



HAL
open science

Contributions au déploiement des services mobiles et à l'analyse de la sécurité des transactions

Vincent Alimi

► **To cite this version:**

Vincent Alimi. Contributions au déploiement des services mobiles et à l'analyse de la sécurité des transactions. Cryptographie et sécurité [cs.CR]. Université de Caen, 2012. Français. NNT: . tel-01007678

HAL Id: tel-01007678

<https://theses.hal.science/tel-01007678>

Submitted on 17 Jun 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



UNIVERSITÉ de
CAEN/BASSE-NORMANDIE
U.F.R. de Sciences
École doctorale S.I.M.E.M

THÈSE

présentée par

M. Vincent Alimi

et soutenue

le 18 décembre 2012

en vue de l'obtention du

Doctorat de l'Université de Caen Basse-Normandie
Spécialité : Informatique et applications

Arrêté du 7 août 2006

Contribution au déploiement des services mobiles et à l'analyse de la sécurité des transactions

MEMBRES du JURY

Jean-Louis Lanet	Professeur à l'Université de Limoges	(Rapporteur)
Gilles Grimaud	Professeur à l'Université de Lille 1	(Rapporteur)
Gildas Avoine	Professeur à l'Université de Louvain (Belgique)	(Examineur)
Pierre Paradinas	Professeur au CNAM	(Examineur)
Bastien Latgé	Ingénieur à INSIDE Secure	(Examineur)
Christophe Rosenberger	Professeur à l'ENSICAEN	(Directeur de thèse)
Sylvain Vernois	Ingénieur de recherche à l'ENSICAEN	(Co-encadrant de thèse)

A ma mère, mon père et mon beau-père, partis trop tôt ...

Résumé

Avec l'avènement de l'Internet grand public et des réseaux numériques, de nouvelles transactions ont vu le jour. Ces transactions, dites « électroniques », sont désormais omniprésentes et régissent notre vie quotidienne : accès par badge à un bâtiment, envoi d'un SMS ou d'un courriel, paiement sur internet, retrait d'argent à un distributeur, ... Les transactions électroniques ont ouvert la voie à une multitude de possibilités déclinées sous diverses formes : le portail internet d'une banque pour consulter ses comptes, effectuer des virements ou passer des ordres en bourse ; une carte à puce permettant d'ouvrir une porte ou de valider son titre de transport en commun ; une application téléchargée sur un ordinateur ou un équipement mobile comme un assistant personnel numérique ou un téléphone portable. Cette dernière catégorie d'équipement mobile est extrêmement porteuse en terme d'offres de services. En effet, un mobile est un équipement nomade, connecté à l'Internet avec des débits de plus en plus élevés, et il est aussi de plus en plus puissant.

Avec l'avènement de la technologie NFC (*Near Field Communication*) et des architectures sécurisées, le mobile a maintenant la possibilité d'héberger des applications sensibles dans une puce sécurisée. Cette puce peut être la carte SIM, une puce embarquée soudée à l'électronique du mobile et une carte mémoire amovible sécurisée. Cette puce sécurisée est appelée « Secure Element ». Contrairement aux facteurs de formes plastiques, le Secure Element situé dans un mobile est un environnement dynamique sur lequel on peut charger des applications qui seront accessibles par un lecteur comme un terminal de paiement, un lecteur de badge d'accès à un bâtiment, un validateur de titre de transport, ... On parle alors de *service mobile sans-contact* ou de *service mobile NFC*. Le déploiement des services mobiles sans-contact s'appuie principalement sur une architecture contrôlée. Lorsque le rôle de « gestionnaire de services de confiance » (*Trusted Service Manager, TSM*) est tenu par l'opérateur de télécommunications, le service est généralement déployé sur la carte SIM dont il est l'émetteur et dont il a une parfaite maîtrise et connaissance. Il réalise alors le

déploiement selon des mécanismes standardisés et éprouvés. En revanche, lorsque le fournisseur de services fait appel à une entité tierce tenant le rôle de TSM, celui-ci doit déployer un service sur un Secure Element dont il n'est pas l'émetteur et donc n'a pas connaissance de sa configuration. Le modèle actuel repose sur une connaissance *a priori* des caractéristiques du Secure Element. Ce modèle empirique a une limitation : lors de déploiements à grande échelle, lorsque plusieurs générations de SE se retrouveront déployées, le TSM devra gérer un parc hétérogène de SE avec une multitude de configurations, de versions logicielles différentes, des fonctionnalités différentes et un contenu applicatif différent...

Cette thèse a pour objectif d'apporter plusieurs contributions pour faciliter le déploiement de services mobiles exploitant un SE. L'adoption de ce type de service pour des applications à forte valeur ajoutée passe par une infrastructure et des outils partagés par les différents acteurs. Dans un premier temps, nous proposons trois contributions pour l'aide au déploiement de services mobiles à base de SE afin de : faciliter la personnalisation d'un SE par un TSM non propriétaire de celui-ci, faciliter l'échange de clés pour le TSM et réaliser une transaction avec un téléphone mobile comme TPE (Terminal de paiement électronique). Dans un second temps, nous nous intéressons à l'analyse de transactions à l'aide d'applications utilisant un SE à des fins de vérification fonctionnelle et sécuritaire.

Table des matières

1	Introduction	1
	Introduction	1
1.1	Contexte de réalisation de la thèse	1
1.2	Motivations	2
1.2.1	Architectures de déploiement	3
1.2.2	Le besoin de sécurité	4
1.2.3	Les services mobiles sans-contact	5
1.2.4	Le déploiement des services mobiles sans-contact	5
1.2.5	Analyse des transactions de services mobiles sans contact	7
1.3	Organisation du manuscrit	8
2	Positionnement du problème	9
2.1	Introduction	9
2.2	Qu'est-ce qu'une transaction électronique ?	10
2.3	Les transactions sans contact	11
2.4	Les services mobiles sans contact	11
2.5	Un service sensible : le paiement mobile	11
2.5.1	Introduction	11
2.5.2	Description	12
2.5.3	Les technologies sans-contact	13
2.5.4	Avantages et inconvénients	18
2.5.5	Discussion	20
2.6	Le « Secure Element »: la réponse au besoin de sécurité	22
2.6.1	Définition et propriétés attendues	22
2.6.2	Plateformes existantes	23
2.6.3	Comparaison et discussion	30

2.7	La technologie NFC couplée à un SE GlobalPlatform	31
2.7.1	L'écosystème NFC	31
2.7.2	Le cycle de vie des services mobiles sans-contact	32
2.8	Conclusion	33
3	Contributions au déploiement et à l'acceptation des services mo- biles sans contact	35
3.1	Introduction	36
3.2	Un mécanisme de distribution à la demande des clés des Secure Element	39
3.2.1	Introduction	39
3.2.2	Description du problème	39
3.2.3	Propositions pour l'identification du tiers de confiance	40
3.2.4	Architecture proposée	42
3.2.5	Cinématique	42
3.2.6	Validation	44
3.2.7	Discussion	45
3.3	Proposition d'un terminal de paiement électronique sur téléphone mobile	45
3.3.1	Introduction	46
3.3.2	Propriétés attendues	46
3.3.3	Les certifications	46
3.3.4	État de l'art des terminaux de paiement sans contact	51
3.3.5	Un terminal de paiement électronique sur un téléphone NFC .	53
3.3.6	Analyse des solutions proposées	56
3.3.7	Discussion	58
3.4	Aide au déploiement de services mobiles sans contact	60
3.4.1	Introduction	60
3.4.2	État de l'art des méthodes de modélisation	61
3.4.3	Présentation du framework	70
3.4.4	Présentation des outils	73
3.4.5	Modélisation en couches du SE	75
3.4.6	Application à la conception d'un outil pour l'aide au déploiement	81
3.4.7	Validation de l'approche	84
3.4.8	Discussion	92
3.5	Conclusion	92
4	Plateforme d'analyse de la sécurité des transactions de paiement	95
4.1	Introduction	96
4.2	Le protocole EMV	96

4.3	Etat de l'art	103
4.3.1	Les méthodes de test de pénétration	103
4.3.2	Attaques sur les cartes à puce et les transactions électroniques de paiement	104
4.3.3	Plateformes d'analyse de transactions électroniques	107
4.3.4	Discussion	108
4.4	Le framework « WinSCard Tools »	109
4.4.1	Introduction	109
4.4.2	Présentation de « WinSCard Tools »	111
4.4.3	Application à l'analyse des transactions	116
4.5	Un framework de fuzzing pour les services de paiement mobile	123
4.5.1	Découverte de vulnérabilités par fuzzing	123
4.5.2	Architecture de fuzzing dans WinSCard Tools	126
4.5.3	1 ^{ère} approche : fuzzing sans implémentation de référence	130
4.5.4	2 ^{nde} approche : fuzzing avec une implémentation de référence	138
4.5.5	Discussion	153
4.6	Conclusion	153
	Conclusions et perspectives	155
	Bibliographie	159
	Annexe	173
A	Scripts de déploiement d'un service mobile sans contact	175
A.1	Récupération des données de configuration du SE – Données vierges	175
A.2	Chargement de l'ontologie génération automatique et envoi du script de déploiement	176
A.3	Mise à jour des données de configuration dans l'applet	178
A.4	Récupération des données de configuration depuis l'applet chargement dans le framework et requête identique de déploiement	181
B	Ontologie OWL GlobalPlatform	187
B.1	L'ontologie GlobalPlatform	187
B.2	Représentation graphique de l'ontologie GlobalPlatform	187
C	Implémentation de référence	189
C.1	Implémentation de référence de l'application MasterCard Paypass Magstripe	189

Liste des abréviations	191
Table des figures	193
Liste des tableaux	197

Publications de l'auteur

Brevets

1. Vincent ALIMI. "Procédé et système pour exécuter une transaction sans contact autorisant de multiples applications et de multiples instances d'une même application". Brevet FR1158943. 4 octobre 2011.

Chapitre de livre international

2. Rima BELGUECHI, Vincent ALIMI, Estelle CHERRIER, Patrick LACHARME, Christophe ROSENBERGER (2011). "An Overview on Privacy Preserving Biometrics". Dans : *Recent Application in Biometrics*. Sous la dir. de Jucheng Yang et Norman Poh. InTech, juil. 2011. Chap. 4, p. 65-84. DOI : 10.5772/19338. URL : <http://www.intechopen.com/books/recent-application-in-biometrics/an-overview-on-privacy-preserving-biometrics>

Article de revue internationale avec comité de rédaction

3. Vincent ALIMI, Christophe ROSENBERGER, Sylvain VERNOIS. "A Mobile Contactless Point of Sale Enhanced by the NFC and Biometric Technologies". Dans : *INDERSCIENCE International Journal of Internet Technology and Secured Transactions*. [accepté pour publication].

Communications avec actes dans un congrès international

4. Vincent ALIMI, "An Ontology-Based Framework to Model a GlobalPlatform Secure Element". Dans : *Proceedings of the 4th International Workshop on Near Field Communication*. Helsinki, Finlande. IEEE Computer Society, 2012, p. 25-30. DOI : 10.1109/NFC.2012.13.
5. Johann VINCENT, Vincent ALIMI, Aude PLATEAUX, Chrystel GABER, Marc PASQUET. "A Mobile Payment Evaluation Based on a Digital Identity Representation". Dans : *IEEE International Symposium on Collaborative Technologies and Systems (CTS)*. Denver, Colorado, USA, mai 2012, p. 410-418. DOI : 10.1109/CTS.2012.6261085.
6. Jean-Claude PAILLÈS, Chrystel GABER, Vincent ALIMI, Marc PASQUET. "Payment & privacy : a key for the development of NFC mobile". Dans : *IEEE International Symposium on Collaborative Technologies and Systems (CTS)*. Chicago, Illinois, USA, mai 2010, p. 378-385. DOI : 10.1109/CTS.2010.5478490.
7. Vincent ALIMI, Marc PASQUET. "Post-distribution provisioning and personalization of a payment application on a UICC-based Secure Element". Dans : *Proceedings of the 1st International Workshop on Sensor Security*. Taux d'acceptation : 25/100. Fukuoka, Japon, mars 2009, p. 701-705. DOI : 10.1109/ARES.2009.98.
8. Marc PASQUET, Sylvain VERNOIS, Vincent ALIMI, Christophe ROSENBERGER. "An e-banking platform for collaborative work between education, industry and research". Dans : *IEEE International Symposium on Collaborative Technologies and Systems (CTS)*. Irvine, Californie, USA, mai 2008, p. 95-102. DOI : 10.1109/CTS.2008.4543918.

Communications orales sans actes dans un congrès international ou national

9. Vincent ALIMI. "A Secure Element Configuration Support System for The Deployment of NFC Mobile Services". Dans : *Chip-to-Cloud Security Forum*. Sophia-Antipolis, France, sept. 2012.
10. Vincent ALIMI. "An ontology-based framework to model a GlobalPlatform Secure Element". Dans : *WIMA NFC Monaco*. Monaco, Principauté de Monaco, avril 2012.

11. Vincent ALIMI, Christophe ROSENBERGER, Sylvain VERNOIS. "A mobile contactless point of sale enhanced by the NFC technology and a match-on-card signature verification algorithm". Dans : *Smart Mobility Event in NFC World Congress Conference*. Sophia-Antipolis, France, sept. 2011.

Communications avec actes dans un congrès national

12. Benoît VIBERT, Vincent ALIMI, Sylvain VERNOIS. "Analyse de la sécurité de transactions à puce avec le framework WinSCard Tools". Dans : *7ième conférence sur la sécurité des architectures réseaux et systèmes d'information (SAR-SSI)*. Cabourg, France, mai 2012, p. 8. OAI : hal.archives-ouvertes.fr :hal-00697465.
13. Vincent ALIMI, Rima BELGUECHI, Christophe ROSENBERGER. "Secure and Privacy Preserving Management of Biometric Templates". Dans : *Third Norsk Information security conference (NISK)*. Gjøvik, Norvège, nov. 2010, p. 134-146. URL : <http://tapironline.no/last-ned/362>.
14. Sylvain VERNOIS, Vincent ALIMI. "WinSCard Tools : a software for the development and security analysis of transactions with smartcards". Dans : *The third Norsk Information security conference (NISK)*. Gjøvik, Norvège, nov. 2010, p. 69-81. URL : <http://tapironline.no/last-ned/356>.

Rapports de recherche

15. Vincent ALIMI. Etat de l'art de la sécurité de la carte à puce, Mai 2009.

Cours

16. Vincent ALIMI. Spécifications EMV, 2012.

Autres publications

17. Vincent ALIMI, Robert BROWN, Laurence BRINGER, Virginie GALINDO, Michele STRUVAY, Sébastien TAVEAU, Mark KEKICHEFF. "A new model : the consumer-centric model and how it applies to the mobile ecosystem". Global-Platform. mars 2012.

Chapitre 1

Introduction

Sommaire

1.1	Contexte de réalisation de la thèse	1
1.2	Motivations	2
1.3	Organisation du manuscrit	8

« L'imagination est plus importante que le savoir. »

Albert Einstein

1.1 Contexte de réalisation de la thèse

Cette thèse a été financée par le biais d'une convention CIFRE établie entre le laboratoire GREYC et la société INSIDE Secure. J'ai été affecté au service *NFC & Payment Solutions* dirigé par Bernard Vian, et plus particulièrement à l'équipe *Secure Element & Applications* sous la direction de Bastien Latgé. Parmi les nombreuses réalisations professionnelles au sein d'INSIDE Secure durant ces trois années, cette dernière m'a notamment chargé de la représenter au sein des consortiums *GlobalPlatform* et *OSPT Alliance* au sein duquel je co-dirige le *Mobile Working Group*. J'ai également été impliqué lors de revues de documents et spécifications du consortium *EMVCo*.

1.2 Motivations

Les services mobiles

Avec l'avènement de l'Internet grand public et des réseaux numériques, de nouvelles transactions ont vu le jour. Ces transactions, dites « électroniques », sont désormais omniprésentes et régissent notre vie quotidienne : accès par badge à un bâtiment, envoi d'un SMS ou d'un courriel, paiement sur internet, retrait d'argent à un distributeur, ... Voici quelques chiffres :

- en France en 2010, le commerce électronique (achats sur internet) représentait des transactions pour un montant de 31 milliards d'euros [1]
- en France en 2010, 103 milliards de SMS ont été échangés [2]
- en France en 2010, on enregistrait 7 milliards de paiements par carte bancaire pour un montant de 336 milliards d'euros, et 1,5 milliards de retraits au distributeur pour un montant de 115 milliards d'euros [3]

Les transactions électroniques ont ouvert la voie à une multitude de possibilités déclinées sous diverses formes : le portail internet d'une banque pour consulter ses comptes, effectuer des virements ou passer des ordres en bourses ; une carte à puce permettant d'ouvrir une porte ou de valider son titre de transport en commun ; une application téléchargée sur un ordinateur ou un équipement mobile comme un assistant personnel numérique ou un téléphone portable. Cette dernière catégorie d'équipement mobile est extrêmement porteuse en terme d'offres de services. En effet, un téléphone portable (que l'on appellera « mobile » par la suite) présente les caractéristiques suivantes :

nomade : il est caractérisé par sa mobilité ce qui lui permet de devenir un outil indispensable de la vie de tous les jours [4, 5] ;

connecté : avec la téléphonie de 3^{me} et maintenant de 4^{me} génération [6], le mobile dispose d'un accès internet itinérant à haut débit. Cela permet, entre autres, la consultation de courriels, l'accès à des services web, ... ;

puissant : les mobiles actuels appelés « *smartphones* » embarquent des processeurs de plus en plus puissants, un espace mémoire de plus en plus important et un système d'exploitation de plus en plus performant.

De par les caractéristiques énoncées ci-dessus, le mobile apparaît comme une plateforme idéale pour le déploiement de services à forte valeur ajoutée. Ces services tirent partie de la connectivité du mobile (parfois de ses capacités de géolocalisation) et de sa puissance. On les appelle « services mobiles ». L'accès à ces services impose

l'installation d'applications sur le mobile de l'utilisateur. Ces applications, ont la particularité de ne pas être délivrées à l'utilisateur sur un support figé comme une carte plastique, mais délivrées *a posteriori*. On parle alors de « post-émission » et de « post-personnalisation ». Ces services sont généralement installés par le réseau de l'opérateur (*Over The Air*, OTA) ou par l'Internet (*Over The Internet*, OTI).

1.2.1 Architectures de déploiement

Le déploiement OTA des services mobiles peut se faire selon 3 modèles principaux : une architecture ouverte, fermée ou contrôlée.

1.2.1.1 Architecture ouverte

L'architecture ouverte de déploiement est typiquement utilisée avec les mobiles de type Android. Sur cette plateforme, il n'y a pas de contrôle particulier des services mobiles développés. Ils peuvent être déployés depuis un ordinateur personnel ou un site web. Même si la plateforme requiert que chaque application déployée soit signée, il s'agit plus de pouvoir établir un lien de confiance entre les différentes applications d'un même auteur que d'autoriser ou non l'installation et l'exécution du service. Il est d'ailleurs autorisé et assez typique pour des applications d'utiliser des certificats auto signés [7].

1.2.1.2 Architecture fermée

Ce type d'architecture est parfaitement illustré par le modèle adopté par Apple[©]. Dans ce modèle, seul Apple[©] délivre l'environnement de développement, le certificat (payant) pour signer les applications, contrôle le code et autorise la diffusion par le biais de l'App Store. Tout ceci est réalisé au travers d'une application sur le mobile ou un ordinateur personnel auquel le mobile est connecté : iTunes.

Ce modèle fermé a été créé pour garantir la sécurité et l'intégrité des équipements de la marque Apple[©]. Cependant, ce système est cassé à chacune de ses mises à jour. On appelle ces attaques des « *jailbreaks* ». Ce déverrouillage permet d'installer des applications non approuvées par Apple[©], ou encore de faire fonctionner le mobile sur le réseau d'un autre opérateur. Le 26 juillet 2010, la DCMA (*Defense Contract Management Agency*) a décidé d'autoriser le déverrouillage des mobiles uniquement dans les cas d'installations d'applications non approuvées ou de changement d'opérateur [8].

1.2.1.3 Architecture contrôlée

Ce type d'architecture ressemble à l'architecture ouverte à l'exception que l'application doit être « certifiée » par une autorité par le biais d'une signature électronique. Les deux principaux cas d'usage de ce type d'architecture sont les suivants :

- la signature des applications mobiles pour leur attribuer des permissions. Par exemple, dans le langage Java pour mobile (J2ME), en fonction du type de certificat délivré, l'application est assignée à un domaine de permissions. Ce domaine détermine les actions et accès aux ressources autorisées [9] ;
- la signature des applications destinées à une carte à puce embarquée (carte SIM ou autre). En fonction de la configuration de la carte à puce, par exemple la carte SIM, l'application doit être signée par l'émetteur de la SIM *i.e.* l'opérateur de télécommunications. La signature est alors vérifiée par la SIM avant d'installer l'application dans un espace appelé domaine de sécurité.

Ces architectures font appel à un rôle particulier appelé « gestionnaire de services de confiance », en anglais *Trusted Service Manager* (nous utiliserons par la suite l'acronyme *TSM*). En fonction du cas d'utilisation et de la configuration de la plateforme cible, le rôle de TSM peut être tenu par des acteurs différents. Il peut être tenu par l'opérateur de télécommunications lui-même ou par un acteur dédié qui s'appuiera sur l'infrastructure de télécommunications de l'opérateur.

1.2.2 Le besoin de sécurité

Certains services mobiles ont besoin d'installer, de manipuler ou de stocker des données dites sensibles. La mémoire du mobile offre un moyen de persistance aux données du service et peut, en conjonction de méthodes cryptographiques, assurer un niveau de sécurité suffisant pour certains services (exemple : application de fidélité). Pour d'autres services en revanche, ce niveau sécurité n'est pas suffisant. Par exemple, dans le domaine du paiement, il est requis que les applications soient hébergées par un composant électronique répondant au niveau de sécurité *Critères Communs EAL5+*. Ce composant est appelé *élément de sécurité*, en anglais *Secure Element (SE)*. Ce SE existe déjà sous diverses formes : une carte à puce à contact (exemple : la carte bancaire plastique) ou une carte à puce sans contact. Le système d'exploitation du SE est basé sur un langage ouvert tel que Java Card™ ou MULTOS™. Basée sur ces langages, une plateforme a été développée par les industriels du domaine de la carte à puce afin de garantir la sécurité du chargement et de la gestion du cycle de vie des applications. Cette plateforme, initiative de Visa©, s'est d'abord appelée OpenPlatform et porte maintenant le nom de GlobalPlatform [10].

1.2.3 Les services mobiles sans-contact

Historiquement, sur les mobiles GSM, la carte SIM était considérée comme le SE par excellence car, en dehors de son usage pour les télécommunications, elle présente les caractéristiques nécessaires. Avec l'avènement du NFC (*Near Field Communication*, cf. 2.5.3.4) et des architectures sécurisées, on compte trois autres implémentations possibles pour le SE [11] : le micro-processeur du mobile, une puce embarquée dans le mobile et une carte mémoire amovible sécurisée. Dans les architectures sécurisées NFC, le SE est utilisé pour émuler une carte sans contact avec le mobile et pour être le garant d'un niveau de sécurité conforme aux Critères Communs EAL5.

Contrairement aux facteurs de forme plastiques, le SE situé dans un mobile est un environnement dynamique [12]. Il permet de charger, via le réseau de l'opérateur, des applications qui seront accessibles par un lecteur comme un terminal de paiement, un lecteur de badge d'accès à un bâtiment, un validateur de titre de transport, ... On parle alors de *service mobile sans-contact* ou de *service mobile NFC*.

1.2.4 Le déploiement des services mobiles sans-contact

Le déploiement des services mobiles sans-contact s'appuie principalement sur une architecture contrôlée. Lorsque le rôle de TSM est tenu par l'opérateur de télécommunications, le service est généralement déployé sur la carte SIM dont il est l'émetteur et dont il a une parfaite maîtrise et connaissance. Il réalise alors le déploiement selon des mécanismes standardisés et éprouvés.

En revanche, lorsque le fournisseur de services fait appel à une entité tierce tenant le rôle de TSM, celui-ci doit déployer un service, sur un SE par exemple, dont il n'est pas l'émetteur et donc n'a pas connaissance de sa configuration. Le modèle actuel repose sur une connaissance *a priori* des caractéristiques du SE. Ce modèle empirique a une limitation : lors de déploiements à grande échelle, lorsque plusieurs générations de SE seront déployées, le TSM devra gérer un parc hétérogène de SE avec une multitude de configurations, de versions logicielles différentes, des fonctionnalités différentes et un contenu applicatif différent...

Prenons l'exemple d'un utilisateur souhaitant effectuer des paiements de proximité avec son mobile doté de la technologie NFC. Celui-ci doit aller à sa banque afin que l'ordre d'installation et/ou personnalisation de sa carte bancaire sur son mobile soit donné. Le banquier va alors se connecter à un outil d'administration et renseigner différentes informations telles que l'identité du porteur, son numéro de téléphone mobile et le type de carte qui doit être déployée sur son équipement. Cette interface

est celle du TSM et le fait de valider l'ordre de déploiement va engendrer une série d'actions consécutives.

- Tout d'abord, le TSM va vérifier l'« éligibilité » de l'équipement. Le mobile est-il compatible avec le service ? Le SE a-t-il des capacités NFC ? Le SE est-il accessible OTA ? Cette vérification est effectuée par le TSM lui-même, auprès du fournisseur du SE ou auprès de l'opérateur.
- Le TSM vérifie ensuite que l'abonnement souscrit par l'utilisateur auprès de son opérateur de télécommunications est compatible avec des opérations de déploiement de services mobiles sans contact.
- Après avoir déterminé s'il détient les privilèges nécessaires pour effectuer les opérations de gestion du SE, le TSM pourra potentiellement réaliser les trois options suivantes :
 - il délègue le déploiement à un autre TSM ;
 - il génère les commandes requises pour le déploiement du service. On appelle alors l'ensemble des commandes un « script » de déploiement ;
 - lorsque le mode de gestion du SE est par délégation alors le TSM doit demander à l'émetteur une preuve que les opérations de déploiement sont autorisées. On appelle cette preuve un « jeton » (*token*).

Cet exemple illustre les limitations du système actuel énoncées précédemment. Certes, il existe des mécanismes permettant au TSM de connaître la configuration du SE mais ils sont basés sur des échanges avec l'émetteur, le fournisseur, l'opérateur de télécommunications ou directement le SE. Typiquement, les SE vendus actuellement sur le marché sont livrés avec des applications pré-chargées qu'il suffit d'instancier pour les rendre active. C'est généralement le cas des applications bancaires comme Visa, MasterCard, ... Mais que se passe-t-il si ce n'est pas le cas ? Comment le TSM a-t-il la connaissance des applications pré-chargées et s'il doit juste instancier l'application ou charger tout le code binaire sur le SE ? Actuellement, ces questions sont résolues par des accords passés entre les acteurs. Ces accords permettent d'établir à l'avance la connaissance de ce qui sera présent sur le SE.

Ces procédés répondent pour l'instant au besoin d'un marché naissant mais ne semble pas pouvoir facilement s'adapter à un marché comptant bientôt des millions voire des milliards de puces. Dans cette thèse, nous nous attachons à définir une méthode qui permet de pallier à ces inconvénients. La modélisation de la plateforme GlobalPlatform permettrait d'établir un langage commun pour décrire un SE implémentant celle-ci. Ainsi, en injectant dans le modèle la description des éléments constituant un SE donné dans ce langage, le TSM obtiendrait une modélisation

du SE sur lequel il souhaite déployer un service. Afin d'aider le déploiement, le modèle pourrait générer lui-même les commandes à envoyer au SE. En effet, le modèle ayant la connaissance de la configuration du SE, du mode de gestion adopté, des fonctionnalités supportées, des contraintes supplémentaires liées au domaine d'exploitation (paiement, fidélité, transport, ...) et du contenu applicatif, il est à même de déterminer le scénario adapté et de générer les commandes que le TSM enverra au SE.

1.2.5 Analyse des transactions de services mobiles sans contact

De par leur nature, les transactions des services mobiles sans contact sont plus sujettes aux attaques de type *man-in-the-middle*, *eavesdropping* et en relais. La découverte de nouvelles vulnérabilités ou de nouveaux types d'attaques passe par la reproduction de celles de l'état de l'art. La reproduction de ces attaques de manière logicielle permettrait de s'affranchir d'un dispositif matériel parfois difficile à mettre en œuvre (exemple : attaque de Cambridge, dans la section 4.3.2.3).

Lors du développement d'applications de services mobiles NFC, les erreurs d'implémentation sont la plupart du temps détectées. Il se peut néanmoins que certaines erreurs ne le soient pas, ce qui pourrait avoir des conséquences désastreuses. Une méthodologie de tests basés sur les techniques de *fuzzing* pourrait aider les développeurs à réaliser des tests fonctionnels et sécuritaires complets à moindre coût.

Contributions

Cette thèse se situe dans la problématique des services mobiles sans contact et des transactions électroniques sécurisées comme le paiement, la fidélité, le transport ou le contrôle d'accès. Un des enjeux de demain est de déployer les services à l'utilisateur actuellement offerts sur une carte à puce, sur un équipement mobile, offrant ainsi plus de capacités d'interaction (écran, clavier) avec l'utilisateur et de flexibilité (ajout d'application, suppression d'application ...). Là encore, l'enjeu est conséquent. Par exemple, en 2010 en France, le taux de pénétration des communications électroniques et services mobiles était de 97% [13].

L'usage d'un équipement mobile dans le cadre de ce type de transactions fait intervenir de nouveaux acteurs qui se limitaient avant à un seul émetteur, un seul porteur, un accepteur et un acquéreur. Les transactions font maintenant intervenir l'opérateur de télécommunications, le fabricant de l'équipement mobile et de la puce sécurisée qu'il contient, ou encore un tiers de confiance (TSM) [14, 15]. Ils contribuent de façons différentes aux architectures de l'état de l'art.

Au cours de cette thèse, nous nous attachons à bien positionner la problématique des services mobiles en termes de méthodes de génération de scripts de déploiement et d'analyse des transactions durant la phase d'utilisation. Nous modélisons la plateforme GlobalPlatform, développons une plateforme capable de générer automatiquement les scripts de déploiement d'un service mobile sans contact par réflexivité et raisonnement, et établissons des éléments de validation du modèle et des scripts générés par l'outil. Nous développons également une plateforme destinée à aider les développeurs à réaliser les tests fonctionnels des applets pendant la phase de développement. Cette même plateforme peut également être utilisée pendant la phase d'utilisation du service mobile sans contact afin d'analyser et d'évaluer la sécurité de la transaction.

1.3 Organisation du manuscrit

Le manuscrit est articulé de la façon suivante :

- **Le chapitre 2** positionne les travaux de cette thèse autour des services mobiles sans contact, le besoin de sécurité et la réponse que constitue le SE.
- **Le chapitre 3** concerne la proposition d'une méthodologie d'aide au déploiement de services mobiles sans contact dans le cadre d'une architecture contrôlée.
- **Le chapitre 4** s'attache à proposer une méthodologie permettant de réaliser des tests fonctionnels approfondis et sécuritaires d'applications embarquées dans le SE.

Enfin, nous concluons ce mémoire et donnons quelques perspectives.

Chapitre 2

Positionnement du problème

Cette thèse adresse les problématiques du déploiement des services mobiles sans contact et de la sécurité des transactions. Ce chapitre définit les différents termes utilisés et notions abordées dans la suite de ce manuscrit. Dans un premier temps, nous présentons la définition d'un service mobile sans contact. Par la suite, nous abordons les différentes architectures et le besoin de sécurité auxquels répond le SE. Enfin, nous présentons la complexité de l'écosystème et les enjeux liés à l'adoption massive de la technologie NFC.

Sommaire

2.1	Introduction	9
2.2	Qu'est-ce qu'une transaction électronique ?	10
2.3	Les transactions sans contact	11
2.4	Les services mobiles sans contact	11
2.5	Un service sensible : le paiement mobile	11
2.6	Le « Secure Element »: la réponse au besoin de sécurité	22
2.7	La technologie NFC couplée à un SE GlobalPlatform	31
2.8	Conclusion	33

2.1 Introduction

AVEC l'explosion du marché de la téléphonie mobile, on a vu apparaitre une multitude de services notamment sur la dernière génération de terminaux appelés les "smart phones", traduit littéralement comme "téléphones intelligents". Le

mobile, outil quasi indispensable de la vie quotidienne, devient un outil de convergence de services, souvent comparé à un "couteau suisse" [16]. Nous définissons dans le cadre de ce chapitre les différents concepts utilisés dans cette thèse.

2.2 Qu'est-ce qu'une transaction électronique ?

Une transaction électronique est un échange d'informations dématérialisées entre deux entités (individus ou organisations) via des systèmes informatiques. Elle implique souvent des échanges d'informations personnelles et confidentielles et doit par conséquent être impérativement sécurisée pour aboutir à une Transaction Électronique Sécurisée (TES) [17]. Une telle transaction est illustrée à la figure 2.1.

Dans [18], S. Bresson analyse ces transactions du point de vue de l'utilisateur et des télécommunications. Du point de vue de l'utilisateur, une transaction électronique est un ensemble cohérent d'échanges d'informations relatives à une même idée ou à un même acte, entre deux terminaux au travers d'un réseau. Du point de vue des télécommunications, une transaction électronique utilisateur peut se décomposer en plusieurs transactions qui composent un ensemble cohérent d'échanges d'informations relatifs à une même idée ou à un même acte au travers d'un réseau. C'est le contexte qui indique si on se place du point de vue de l'utilisateur ou des télécommunications. Une transaction électronique peut être l'interrogation d'un stock, l'envoi d'une facture, un paiement à distance, un paiement de proximité, l'utilisation d'un badge de contrôle d'accès...

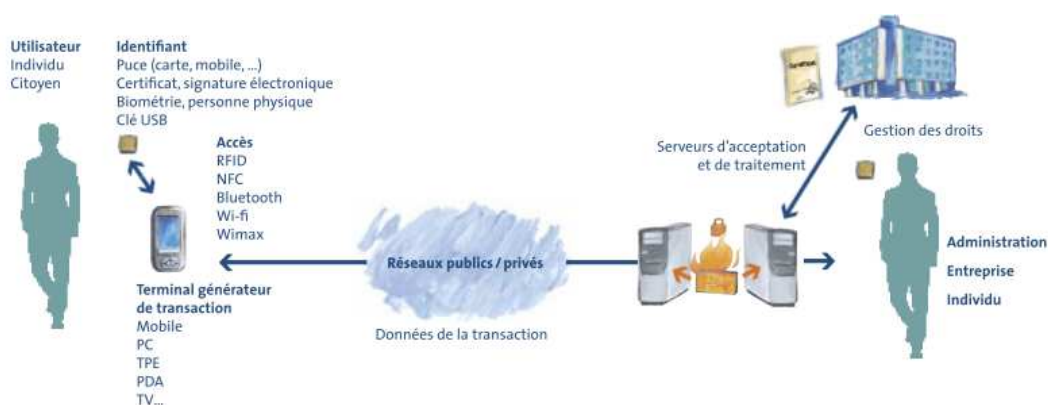


FIGURE 2.1 – Le périmètre de la chaîne d'une transaction électronique sécurisée (source : Pôle Transactions Électroniques Sécurisées [19])

2.3 Les transactions sans contact

Une transaction sans contact est un échange de données réalisé entre deux équipements (doté d'une technologie sans contact). Communément, l'un des équipements est vu comme équipement accepteur. Il existe le modèle maître/esclave dans lequel un équipement est à l'initiative de la transaction et où l'autre ne fait que répondre aux requêtes du premier ; le modèle pair-à-pair où les équipements peuvent prendre tour à tour le rôle de maître et influencer la transaction.

2.4 Les services mobiles sans contact

Les services mobiles sans-contact proviennent de la combinaison, la convergence, de plusieurs technologies. On peut lire dans [20] : « La technologie du NFC permet d'effectuer des échanges de données à courte distance entre un mobile et un lecteur sans contact. Appliquée au téléphone mobile, elle peut donc en faire un portefeuille électronique capable d'offrir des services de paiement, de transport, de fidélité... ». Cette citation illustre bien le courant actuel et le concept sous-jacent des services mobiles sans-contact : accroître de manière conséquente l'offre de services liés à la mobilité par l'utilisation des technologies sans contact. Une récente analyse de marché [21] a montré, suite aux différentes expérimentations menées (notamment à Caen), que le téléphone mobile était le média préféré pour les services à l'utilisateur au dépend des cartes sans contact, autocollants sans contact (stickers RFID) ou autre code barre accroché à un porte-clef. Il présente les avantages suivants : (i) plus pratique car il regroupe différents services dans le même équipement, (ii) plus intégré/développé car il permet plus d'interaction grâce à un clavier et un écran, (iii) évolutif car il permet l'ajout dynamique de nouveaux services.

2.5 Un service sensible : le paiement mobile

2.5.1 Introduction

Parmi les services mobiles sans contact, le paiement est très important car il est générateur de revenus pour les acteurs qui le mettent en œuvre. Nous nous focaliserons donc sur les technologies répondant à cette application particulière et ayant des contraintes : temps de transaction, facilité d'usage, sécurité, ...

2.5.2 Description

Une transaction de paiement mobile peut prendre différentes formes [22, 23, 24]. Il peut s'agir d'un paiement réalisé grâce à des requêtes échangées par le biais de messages SMS (*Short Message Service*) ou USSD (*Unstructured Supplementary Services Data*). Le coût du bien acheté est alors prélevé sur sa facture d'opérateur de services mobiles¹ ou sur un compte détenu auprès du prestataire de services. Un exemple commun est le téléchargement de sonneries ou de fonds d'écran pour son mobile. L'utilisateur envoie alors un SMS à un numéro spécial avec la référence du produit qu'il souhaite voir installé sur son mobile. Le revenu (surtaxé) du SMS est alors partagé entre l'opérateur et le fournisseur de service. Un autre moyen de paiement mobile s'appuie sur la navigation par l'utilisateur sur un site marchand à partir du navigateur de son mobile. Plusieurs options s'offrent alors à l'utilisateur pour effectuer le paiement :

payer par carte bancaire : l'utilisateur saisit les détails de sa carte (le PAN (*Primary Account Number*), la date d'expiration et le cryptogramme visuel au dos de la carte) et le marchand effectue une demande d'autorisation auprès de la banque du porteur ;

être facturé par l'opérateur : le bien est facturé à l'utilisateur sur sa facture de téléphone mobile. Cette option nécessite une intégration de la plateforme de facturation de l'opérateur par le marchand ;

utiliser un compte en ligne chez un opérateur de paiement : des acteurs comme *PayPal*©, *Amazon Payments*© et *Google Checkout*© offrent la possibilité d'ouvrir un « porte-feuille » en ligne et d'y déposer de l'argent.

Une autre famille de paiements mobiles connaît un franc succès depuis quelques années : le paiement mobile de proximité. Ce type de paiement consiste à réaliser une transaction de paiement à l'aide d'une communication à courte distance entre le mobile et le terminal de paiement. Cette communication peut être visuelle, notamment à l'aide d'un code barre comme dans les solutions *Starbucks*©, *PayScan*© et *Cimbal*©, sonore comme la solution *Tagattitude*© ou bien à l'aide d'une technologie radio dite sans-contact. Dans la prochaine section, nous étudions les technologies sans-contact dont peut être équipé un mobile et donnons un exemple d'utilisation pour effectuer un paiement mobile de proximité.

1. Plus connu sous le non anglais de « carrier billing ».

2.5.3 Les technologies sans-contact

2.5.3.1 Le Bluetooth

La technologie Bluetooth[®], standardisée dans l'IEEE 802.15, est une technologie sans-contact utilisée pour connecter plusieurs équipements à courte distance et formant un réseau PAN (*Personal Area Network*) [25]. Ses applications les plus répandues sont l'envoi et la synchronisation de données entre deux équipements, comme un assistant personnel et un ordinateur, et la connexion à Internet [26]. Les spécifications sont maintenues par le *Bluetooth[®] Special Interest Group (SIG)*² créé en 1998 par Ericsson, IBM, Intel, Nokia et Toshiba. La dernière version est la 4.0 qui assure une consommation réduite et un temps de réponse court.

Pour assurer une compatibilité entre tous les périphériques Bluetooth[®], la majeure partie de la pile de protocoles est définie dans la spécification. Elle est représentée sur la figure 2.2. Au niveau de la couche radio, Bluetooth[®] opère dans les bandes de fréquences ISM (*Industrial, Scientific and Medical*) 2,4 GHz dont l'exploitation ne nécessite pas de licence. Cette bande de fréquences est comprise entre 2400 et 2483,5 MHz et offre 79 canaux espacés d'1 MHz [27].

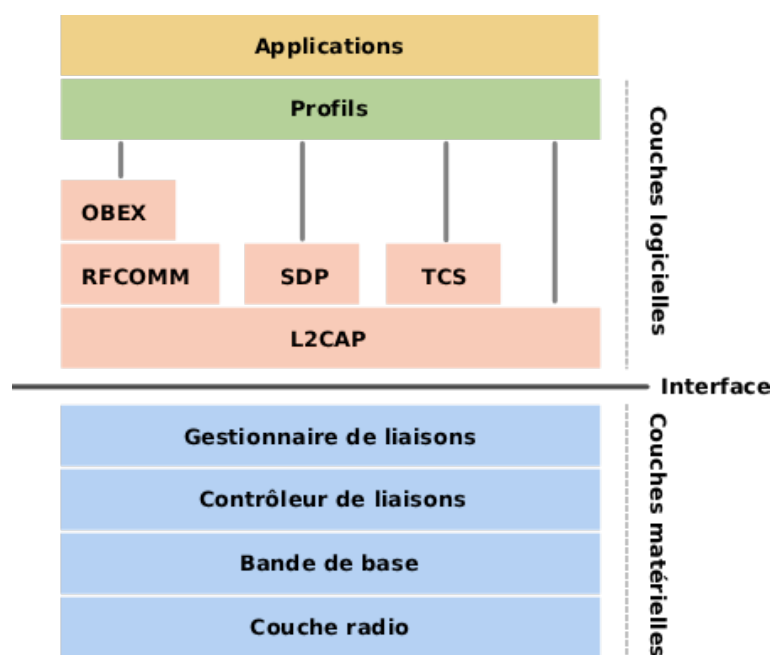


FIGURE 2.2 – Pile de protocole Bluetooth[®]

Bluetooth[®] est basé sur un modèle maître-esclave. Le gestionnaire de liaisons est responsable de l'établissement de la liaison, sa gestion et sa clôture. Il est responsable des mécanismes suivants :

2. <https://www.bluetooth.org>

- l'authentification mutuelle des équipements,
- l'association des périphériques,
- la création et la modification des clés de sécurité,
- le chiffrement de la communication.

Dans [28], Pradhan *et al.* détaillent trois solutions mettant en œuvre la technologie Bluetooth[®] pour une transaction de paiement mobile. Une société française, MobiNear, a développé une technologie basée sur Bluetooth[®] appelée BlueNFC permettant de réaliser des transactions sans contact de proximité avec un mobile (paiement, fidélité, transport, coupons).

2.5.3.2 Le Wi-Fi

La technologie Wi-Fi est un ensemble de protocoles de communication sans fil standardisée dans l'IEEE 802.11. Le Wi-Fi permet de créer des réseaux sans fil WLAN (*Wireless Local Area Network*) afin de transmettre des données entre plusieurs appareils informatiques (ordinateur, routeur, imprimante, ...) sur une liaison à haut débit. Les débits atteignent 11 Mbit/s théoriques pour le Wi-Fi 802.11b, 54 Mbit/s théoriques pour le Wi-Fi 802.11a et 802.11g et 600 Mbit/s théoriques pour le Wi-Fi 802.11n sur une distance de quelques dizaines de mètres.

Sur le modèle OSI, le Wi-Fi définit les couches physiques et liaison de données. La couche physique spécifie la manière dont sont codés et envoyés les bits sur le média. La couche liaison de données définit principalement la méthode d'accès au média et les règles de communication entre les différentes stations [29]. Le Wi-Fi permet plusieurs architectures pour mettre en réseau les « nœuds » :

- *infrastructure* : dans cette architecture, les nœuds se connectent à un point d'accès aussi appelé concentrateur. Tous les nœuds et le point d'accès doivent alors partager le même nom de réseau ou SSID (*Service Set Identifier*). Les points d'accès doivent être placés de manière régulière sur le site afin de couvrir tout l'espace et ne pas risquer de rupture de service.
- « ad-hoc » : cette architecture ne nécessite pas de concentrateur. Les nœuds communiquent deux à deux directement entre eux.
- pont : cette architecture permet à des points d'accès de se connecter à un même autre point d'accès afin d'étendre un réseau filaire. Le point d'accès concentrateur est alors en mode racine et les autres points d'accès s'y connectant sont en mode pont et peuvent ensuite retransmettre la connexion sur leurs interfaces filaires.
- répéteur : un point d'accès en mode répéteur permet de capter puis de répéter un signal radio. Cela peut, par exemple, être utile pour faire parvenir le signal

au fond d'un couloir en L.

Le Wi-Fi offre des services permettant de sécuriser la communication. En effet, le média de communication étant une onde hertzienne, celui-ci est très sensible aux attaques de type *eavesdropping*, *man-in-the-middle* et déni de service par *spoofing* [30, 31, 32]. Les principaux protocoles de chiffrement et d'authentification pour le Wi-Fi sont :

- le WEP (*Wired Equivalent Privacy*) : le protocole WEP, intégré dans IEEE 802.11, utilise l'algorithme symétrique RC4 avec des clés d'une longueur de 64 ou 128 bits. Dans un premier temps, une clé secrète de 40 ou 128 bits est définie au niveau du point d'accès et du nœud dans le réseau. Cette clé est utilisée pour créer un nombre pseudo-aléatoire de même longueur que la trame à envoyer. La trame est alors chiffrée en appliquant un masque OU Exclusif entre la trame en clair et le nombre pseudo-aléatoire.
- le WPA (*Wi-Fi Protected Access*) : le protocole WPA, défini dans la norme IEEE 802.11i, vise à améliorer les deux faiblesses identifiées du protocole WEP : le chiffrement des données et l'authentification des utilisateurs [33]. WPA s'appuie sur TKIP (*Temporal Key Integrity Protocol*). TKIP utilise une fonction de mixage de clés par paquet, un contrôle d'intégrité des messages permettant de détecter une altération et un vecteur d'initialisation étendu (passe de 24 à 48 bits) ce qui permet d'augmenter le nombre de clés partagées possibles et d'éviter les attaques par rejeu. TKIP crée une clé temporaire basée sur une clé partagée entre le point d'accès et le nœud, l'adresse MAC du nœud, le vecteur d'initialisation et cette clé est changée tous les 10 000 paquets. L'authentification, quant à elle, est améliorée par l'utilisation d'EAP (*Extensible Authentication Protocol*).

Comme l'a souligné Zmijewska dans [34], le Wifi n'est pratiquement utilisé que pour connecter des ordinateurs à l'internet ou à des périphériques domestiques et, de ce fait, son utilisation dans le cadre d'un paiement mobile ne semble pas se justifier. Cependant, le Wifi peut être utilisé pour rendre les terminaux de paiement portatifs en établissant une liaison sans fil entre le terminal et sa base. Parmi ces solutions, on compte le VeriFone Vx670 et Ingenico i7780.

2.5.3.3 La technologie sans-contact RFID

Dans l'ouvrage faisant référence [35], la technologie sans-contact RFID (Radio Frequency Identification) est présentée comme « un transfert sans-contact de données entre le dispositif de transport de données et son lecteur. L'énergie requise pour

faire fonctionner le dispositif de transport de données est fournie par le champ électromagnétique émis par le lecteur ». On parle alors de couplage par induction (cf. Figure 2.3).

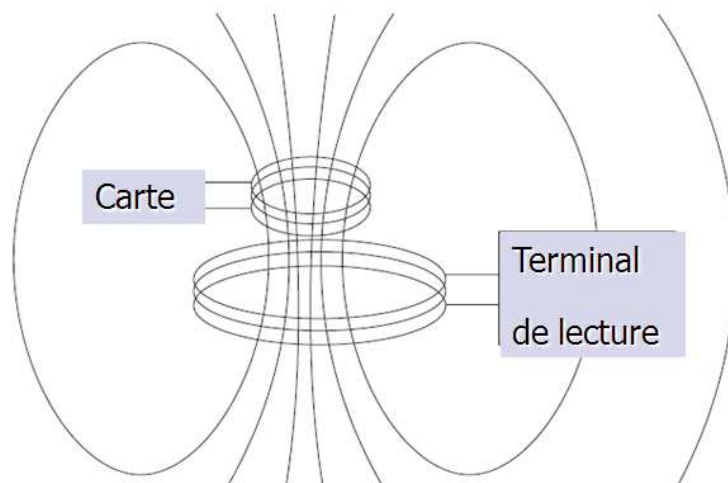


FIGURE 2.3 – Couplage par induction

Le RFID fonctionne à différentes fréquences allant de 135 kHz à 24 GHz. Le transfert de données s’opère par un mécanisme de modulation autour d’une fréquence porteuse (cf. Figure 2.4).

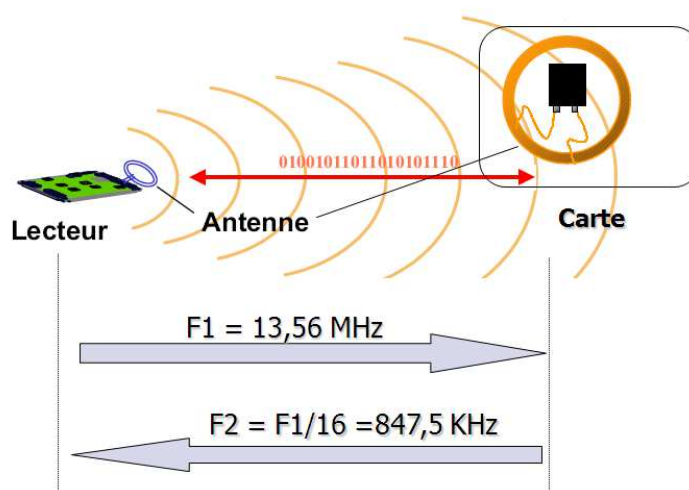


FIGURE 2.4 – Transfert de données par modulation autour d’une fréquence porteuse de 13,56 MHz (ISO 14443)

Une des applications la plus connue des systèmes RFID est le couplage de cartes d’identification, décrite dans plusieurs normes comme les normes ISO 14443 [36] et

ISO 15693 [37]. A titre d'exemple, la norme ISO 14443 est utilisée pour le paiement sans-contact et les transports publics et ISO 15693 pour les badges de contrôle d'accès.

2.5.3.4 Near Field Communication

2.5.3.4.1 Définition

Le NFC (*Near Field Communication*, communication en champ proche) est une technologie basée sur le RFID (Radio Frequency IDentification) qui, embarquée dans un équipement mobile – comme un téléphone portable, une tablette, ... – permet à ce dernier de fonctionner selon trois modes [14] (*cf.* Figure 2.5). Il peut agir comme un lecteur de carte ou de tag sans-contact, communiquer en peer-to-peer³ avec un autre équipement mobile doté de la technologie NFC, ou émuler une carte. Ces trois modes de fonctionnement sont générateurs de services mobiles (échanges de contact en peer-to-peer, lecture d'information sur un « *smart poster* », paiement avec le mobile, ...), mais l'émulation de carte est le mode qui nous intéressera ici. En effet, l'émulation de carte est le mode qui présente l'écosystème le plus complexe et les exigences sécuritaires les plus fortes.



FIGURE 2.5 – Modes de fonctionnement du NFC (source : INSIDE Secure)

3. pair à pair

2.5.3.4.2 Architecture

Un équipement doté de la technologie NFC est composé de trois composants logiques [38, 39] : le processeur de l'équipement mobile (*Baseband Processor*), un contrôleur NFC (*NFC Controller*) et un élément de sécurité (*Secure Element, SE*). Le processeur de l'équipement héberge les applications systèmes de l'équipement (exemple : application de gestion du protocole de télécommunications GSM) et les applications tierces installées par l'utilisateur. Le contrôleur NFC effectue la conversion analogique/numérique des signaux et le routage des communications. Il transmet les messages reçus du monde extérieur au SE et au processeur. L'élément de sécurité (*cf.* section 2.6) permet l'émulation d'une carte à puce avec l'équipement mobile de telle sorte qu'un terminal ne distingue pas une réelle carte sans-contact d'un SE. La figure 2.7 montre l'inter-connexion de ces composants. La figure 2.6 illustre l'intégration du NFC dans un mobile.

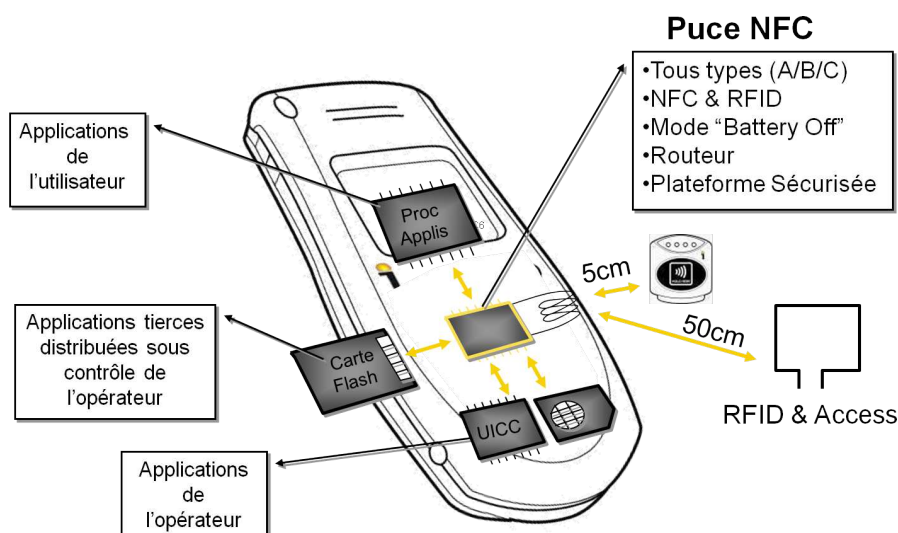


FIGURE 2.6 – Architecture fonctionnelle de la technologie NFC intégrée dans un mobile (source : INSIDE Secure)

2.5.4 Avantages et inconvénients

2.5.4.1 Le Bluetooth

De nos jours, la plupart des mobiles sur le marché sont équipés de la technologie Bluetooth. Comme nous l'avons vu à la section 2.5.3.1, le Bluetooth est un protocole de communication sécurisé. Il offre des services d'authentification mutuelle des péri-

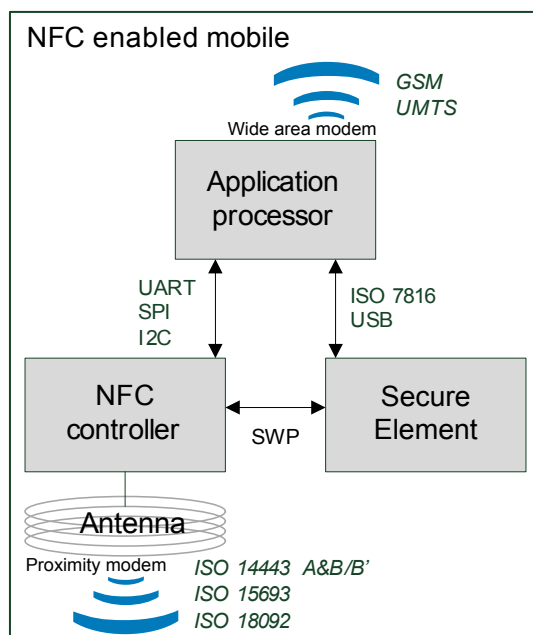


FIGURE 2.7 – Architecture technique du NFC (source : ENSICAEN)

phériques et le chiffrement de la communication. Il s'avère adapté pour la connexion d'un mobile avec des accessoires audio ou d'un ordinateur avec lequel il se synchronise. Cependant, la communication entre les périphériques est précédée d'une procédure d'association entre ceux-ci (pendant laquelle s'opère l'authentification mutuelle) qui s'avère contraignante et longue. Certes, il est possible d'automatiser cette procédure mais elle peut s'avérer être un obstacle lors de transactions ponctuelles et rapides comme un paiement de proximité. De plus, Bluetooth est sensible à de nombreuses attaques telles que les attaques par connaissance du clair chiffré [40], par force brute, déni de service ou *rollback* [41].

2.5.4.2 Le Wi-Fi

Nous avons étudié à la section 2.5.3.2, les différentes architectures du Wi-Fi. Dans le cas d'utilisation qui nous intéresse, seuls les modes *infrastructure* et *ad-hoc* sont significatifs. Tout comme le Bluetooth, les mobiles du marché sont de plus en plus souvent équipés de la technologie Wi-Fi. Elle permet une relative simplicité de mise en œuvre et de connexion avec le point d'accès. Elle offre également des services de sécurité. Le protocole WEP permet le chiffrement de la communication mais a très vite montré des faiblesses. Etant donné la portée du Wi-Fi (de l'ordre de la centaine de mètres) et l'aspect omnidirectionnel de la propagation des ondes radios, il est aisé d'« écouter » les communications et de déchiffrer les trames échangées. Les auteurs

dans [42, 43, 44], détaillent des méthodes pour attaquer le protocole WEP. Des outils en téléchargement tels que WepAttack [45] permettent de réaliser ces attaques chez soi!

Pour palier à ces problèmes de sécurité, le protocole WPA introduit un niveau de sécurité accru avec un changement de clé toutes les 10000 trames chiffrées et un mécanisme d'authentification des périphériques. Cependant, la sécurité du protocole nécessite la mise en place d'une clé à la fois sur le point d'accès et sur le client. Ceci n'est pas envisageable car il s'agit d'une procédure trop longue pour une transaction de paiement. On pourrait répondre à cela que cette procédure ne doit être réalisée qu'une seule fois de manière manuelle et sera automatique toutes les fois d'après, mais si un client détient la clé, il détient la même que celle utilisée par tous les autres clients et peut alors écouter les communications.

2.5.4.3 Le NFC

Le NFC, en mode émulation de carte, permet de réaliser une transaction à courte portée (entre 1 et 10 cm). Certes, dans son utilisation courante, NFC n'est pas sécurisé et est sensible à de nombreuses attaques [46, 39], mais sa courte portée permet justement à l'utilisateur de s'assurer qu'aucun dispositif malveillant ne soit présent entre son mobile et le terminal sans contact du commerçant. Il est à noter qu'il existe tout de même des méthodes de sécurisation de la communication comme énoncés dans [46] et [47]. L'un des points forts de cette technologie est sa rapidité. En effet, le mode émulation de carte a été conçu pour que le mobile se comporte exactement comme une carte sans-contact que ce soit en termes de fonctionnalités que de durée de transaction. Une transaction NFC ne nécessite ni association ni de secret partagé entre les périphériques.

2.5.5 Discussion

2.5.5.1 Quelle technologie sans-contact pour le paiement mobile ?

Le Bluetooth ne semble pas être adapté à une transaction de paiement sans contact de proximité. Le plus grand obstacle semble être la phase d'association des périphériques qui prend trop de temps et n'est pas compatible avec le besoin de rapidité pour le porteur. Dans [48], K. Moyer donne des raisons supplémentaires de l'inadéquation de Bluetooth pour le paiement parmi lesquelles on retrouve les suivantes :

- les commerçants doivent se munir d'un terminal spécifique afin de pouvoir gérer les périphériques Bluetooth ;

- les commerçants ont besoin d’une application pour s’assurer de l’identité du porteur car le terminal risque de s’associer avec n’importe quel mobile situé dans le lieu de vente ;
- un manque général de commodité : l’utilisateur se désintéressera de la technologie s’il doit associer manuellement son mobile avec le terminal du commerçant.

Le Wi-Fi souffre également des mêmes réticences de la part des utilisateurs. Mais le Wi-Fi, en plus du besoin de s’associer à un point d’accès ou à un autre équipement, a une très mauvaise réputation en terme de sécurité des communications. Les logiciels permettant d’attaquer les réseaux Wi-Fi sont nombreux sur le Web. Pour ces différentes raisons, le Wi-Fi ne semble pas être la technologie sans contact la plus appropriée pour le paiement.

Le NFC, quant à lui, offre toutes les caractéristiques nécessaires. En effet, il s’active très facilement sur un mobile, ne nécessite pas d’association avec le terminal et permet des communications rapides. Il souffre certes d’un manque de sécurité lié à l’utilisation du RFID mais la proximité avec le terminal du commerçant assure à l’utilisateur un certain contrôle. Le NFC apparaît comme la meilleure technologie sans contact pour le paiement sans contact de proximité.

Notre choix se trouve conforté par l’étude des technologies sans contact pour le paiement mobile menée par Zmijewska dans [34]. Dans cette étude, les technologies sans contact Bluetooth et NFC sont comparées selon les critères suivants : facilité d’utilisation, utilité, confiance, mobilité et coût. Le NFC se classe premier dans tous les critères sauf le coût car les téléphones équipés du NFC sont généralement des mobiles haut de gamme.

2.5.5.2 Quelle architecture de gestion de services mobiles sans contact ?

Comme nous l’avons vu à la section 2.5.3.4, le composant logique hébergeant les applications qui seront utilisées par un lecteur lorsque le mobile se comporte en émulation de carte est un environnement dynamique sur lequel les applications peuvent être chargées, installées, mises à jour et effacées. Grâce à la connectivité du mobile, ces opérations peuvent être réalisées Over-The-Air (OTA) en *post-personnalisation*. Cela ouvre la voie à un nouvel écosystème dans lequel on retrouve des développeurs d’applications (ex. : développeur d’une application de paiement), des fournisseurs de services (ex. : une banque qui veut proposer un service de paiement mobile de proximité à ses clients) et des plateformes de déploiement de services (ex. : l’opérateur de télécommunications qui installera le service de paiement d’une banque sur les cartes SIM de ses clients). Dans ce contexte, et comme nous l’introduction de ce mémoire,

l'architecture contrôlée semble être le modèle le plus adéquat à la gestion des services mobiles sans contact. Il permet l'ouverture vers les fournisseurs de services tout en gardant le contrôle par l'émetteur.

2.5.5.3 De la sécurité des applications

Les services mobiles sans contact comme le paiement ou le transport s'appuient sur des applications dites sensibles et manipulant des données également sensibles. L'accès frauduleux à ces données ou au code de l'application mettrait en péril non seulement la protection de la vie privée de l'utilisateur mais aussi toute la sécurité du système sur lequel le service repose. Ainsi, des exigences sécuritaires fortes ont été posées sur le composant logique hébergeant les applications. On appelle alors un composant logique répondant à ces exigences un « élément de sécurité » ou « Secure Element ». Les caractéristiques attendues d'un tel composant, les solutions existantes et celle retenue dans le cadre de cette thèse sont détaillées à la section suivante.

2.6 Le « Secure Element » : la réponse au besoin de sécurité

2.6.1 Définition et propriétés attendues

Le SE est un organe de sécurité dans lequel sont hébergées des applications et des données dites « sensibles ». C'est une combinaison de matériels, logiciels et protocoles qui doit offrir une grande résistance aux attaques logiques et physiques. On attend d'un tel composant les propriétés suivantes [12] :

sécurité : cette propriété implique que le SE soit capable de fournir les services d'intégrité, de confidentialité et d'authentification ;

mémoire sécurisée : l'espace mémoire doit présenter un niveau de sécurité suffisant permettant d'y stocker des clés privées et secrètes, certificats racines et données personnelles ;

mécanisme de gestion du contenu : le SE doit permettre la gestion des données et des applications qu'il contient y compris à distance ;

présence de routines cryptographiques : elles doivent notamment permettre l'échange sécurisé de données entre le SE et le monde extérieur ;

exécution sécurisée du code : le SE doit exécuter le code des applications hébergées de manière sécurisée, *i.e.* sans pouvoir être espionné de quelque manière que ce soit.

2.6.2 Plateformes existantes

Au niveau matériel, le SE est implémenté sous la forme d'une puce intégrée selon différents facteurs de forme [49] : la carte SIM d'un mobile (*Universal Integrated Circuit Card, UICC*), une puce embarquée (*embedded SE, eSE*) ou un support amovible sécurisé de type carte mémoire (*Secure Memory Card, SMC*). Enfin, une dernière possibilité consiste à implémenter le SE dans le processeur principal du mobile. Au niveau logiciel, les SEs du type UICC, eSE ou SMC sont basées sur des systèmes d'exploitation ouverts comme Java Card™ [50] et MULTOS™ et répondent aux spécifications GlobalPlatform [51] pour le chargement sécurisé des applications. Lorsque le SE est implémenté dans le processeur du mobile, il peut s'appuyer sur des environnements d'exécution sécurisée (*Trusted Execution Environment, TEE*). Les sections suivantes décrivent ces différentes plateformes.

2.6.2.1 Java Card™

Un système d'exploitation est souvent désigné comme « une plateforme logicielle au-dessus de laquelle d'autres programmes, appelés applications, peuvent être exécutés. Les applications doivent être écrites pour être exécutées par un système d'exploitation particulier » [52].

Au début des années 1990, les prémisses des systèmes d'exploitation pour cartes à puce consistaient en quelques routines « gravées » en ROM (*Read Only Memory*) qui permettaient la communication avec le monde extérieur (conformément à l'ISO 7816-4), des opérations simples sur les fichiers (lecture/écriture de blocs) et quelques opérations cryptographiques. A cette époque, les cartes ne disposaient que de 1-3 Ko de ROM, 128 octets de RAM (*Random Access Memory*) et 1-2 Ko d'EEPROM (*Electrically Erasable Programmable ROM*). La raison principale qui a amené à placer le code en ROM est que cette mémoire utilisait (et utilise toujours) peu de place — et donc un faible coût matériel — par rapport à la taille du microprocesseur [53]. La structure de ces cartes était qualifiée de monolithique car il était impossible de modifier le code en ROM une fois embarqué sur la carte. Pour contourner ce problème, certains concepteurs de puces utilisaient un système de copie de code de la ROM vers l'EEPROM. Quoiqu'il en soit, programmer de la ROM nécessite un cycle long et coûteux qui monopolise beaucoup de compétences. Il fallait donc trouver un moyen rapide pour développer des applications.

La deuxième génération de cartes à puces émergea entre 1995 et 1999 avec l'introduction de microprocesseurs plus puissants (6-8 Ko de ROM, 128 octets de RAM et 6-12 Ko d'EEPROM) et un jeu d'instructions plus étendu. Mais, même si ces cartes présentaient une amélioration, les applications restaient très dépendantes du

système d'exploitation⁴ et de la couche matérielle. La portabilité, le multi-applicatif et la personnalisation post-émission deviennent des problèmes majeurs.

La troisième génération de système d'exploitation vit le jour en février 1997 lorsque Schlumberger et Gemplus annoncent la mise au point de cartes qui embarquent des applications écrites dans le langage Java d'Oracle et publient l'*API Java Card™ v1.0*. Les principales caractéristiques de ce type de système d'exploitation sont :

- interopérabilité des applications,
- isolation entre les applications,
- indépendance vis-à-vis de la couche matérielle (*Hardware Application Layer*),
- sécurité.

On parle alors de plateforme multi-applicative ouverte. La plateforme ouverte la plus répandue est la plateforme Java Card™. Java Card est un environnement d'exécution destiné aux cartes à puce permettant d'écrire et d'exécuter des programmes, appelés *applets*, avec l'approche orientée objet du langage Java. L'architecture Java Card est présentée à la figure 2.8. La technologie Java Card™ propose les avantages suivants :

- portabilité, par abstraction de la couche matérielle sous-jacente ;
- sécurité, par un langage fortement typé et les mécanismes de l'environnement d'exécution ;
- adaptabilité aux environnements de développement Java déjà existant.

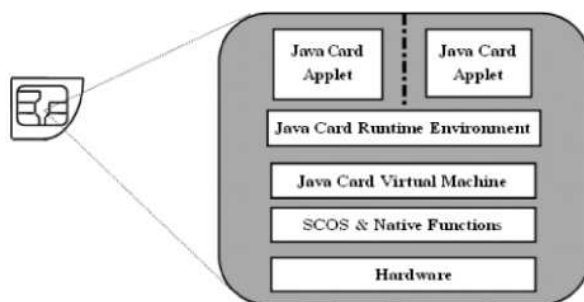


FIGURE 2.8 – Architecture de Java Card™ (source : Addison-Wesley)

2.6.2.2 MULTOS™

Le système d'exploitation MULTOS™ a été spécialement conçu pour les microprocesseurs et avec un soucis constant de sécurité. Un consortium industriel, nommé

4. Quelques exemples de systèmes d'exploitation (SCOS) : MPCOS de Gemplus, STARCOS de Giesecke & Devrient (G&D), CardOS de Siemens, OSCAR de GIS.

MAOSCO, est chargé de le promouvoir, de gérer les spécifications et de fournir les licences et certifications. Les éléments clés de la plateforme MULTOS™ sont :

- une architecture hautement sécurisée,
- l’hébergement de plusieurs applications sur la même carte,
- une couche d’abstraction de la plateforme matérielle,
- la compatibilité avec les standards ISO 7816 et EMV [54] .

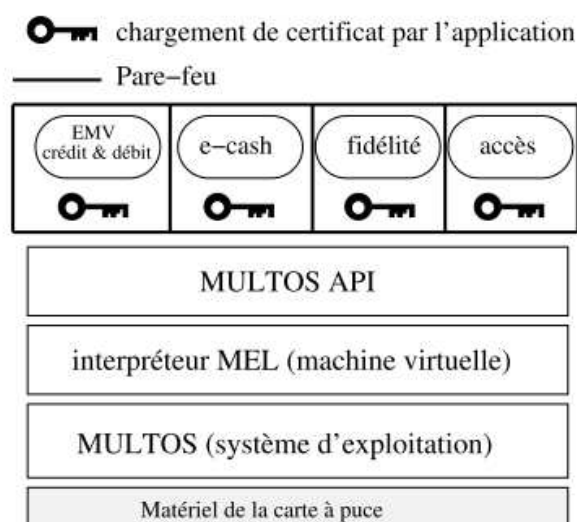


FIGURE 2.9 – Architecture de MULTOS™ (source : Damien Sauveron)

Les applications sont développées dans un langage optimisé pour les cartes à puces : le MEL (Multos™ Executable Language) basé sur les P-codes Pascal et la machine virtuelle décrite dans [55]. Du point de vue de la sécurité, la carte MULTOS™ est la carte la plus sécurisée qui existe puisqu’elle est certifiée au niveau EAL7 des Critères Communs.

2.6.2.3 GlobalPlatform

GlobalPlatform est un consortium créé en 1999 par les grandes entreprises des secteurs du paiement, des télécommunications, et gouvernemental ; et fut le premier à promouvoir une infrastructure globale pour l’implémentation des cartes à puce commune à tous les secteurs industriels. Les spécifications GlobalPlatform (anciennement Visa Open Platform) visent à gérer les cartes de façon indépendante du matériel, des vendeurs et des applications. Elles répondent efficacement aux problématiques de la gestion du multi-applicatif : chargement sécurisé des applications, gestion du contenu, cycle de vie. Les spécifications GlobalPlatform sont divisées en trois thèmes :

- les cartes, qui incluent :
 - *Card Specification v2.2* [51]
 - *Card Specification Amendments A* [56]
- les terminaux, qui incluent :
 - *GlobalPlatform Trusted Execution Environment* [57]
- les systèmes, qui incluent :
 - *GlobalPlatform Messaging Specification v1.0* [58]
 - *Key Management Requirements Systems Functional Requirements Specification v1.0* [59]

2.6.2.3.1 L'architecture GlobalPlatform

L'architecture GlobalPlatform (*cf.* figure 2.10) est composée d'un certain nombre d'éléments logiques et physiques destinés à fournir l'interopérabilité et la sécurité des applications.

Dans les couches basses de l'architecture, on trouve le microprocesseur. Le *Runtime Environment* fournit une API (*Application Programming Interface*), couche indépendante de la couche matérielle, ainsi qu'un espace d'exécution sécurisé pour chaque application afin de leur garantir une séparation du code et des données. Le *Trusted Framework* propose des services de communication sécurisées entre les applications. Les grandes responsabilités de l'environnement d'exécution GlobalPlatform (OPEN) sont de fournir une API aux applications, distribuer les commandes (APDUs reçues), sélectionner des applications et gérer le contenu de la carte.

Les domaines de sécurité sont les représentants sur la carte des acteurs auxquels ils appartiennent. Ils sont les garants du respect de la politique de sécurité que leurs "propriétaires" souhaitent appliquer. Il existe trois types de domaine de sécurité (*Security Domain*, SD) :

- *Issuer Security Domain* (ISD). C'est la première application installée sur la carte. Il est principalement utilisé pour effectuer la gestion du contenu pour l'émetteur de la carte. Par exemple, l'ISD détient les clés émetteur et effectue les calculs cryptographiques lorsque le contenu de la carte change (*e.g.* vérification de *token*).
- *Supplementary ou Application Provider Security Domain*, (SSD ou APSD). C'est un environnement sécurisé dans lequel les fournisseurs d'applications sont autorisés à télécharger, installer des applications et les gérer selon des cycles de vie séparés.
- *Controlling Authority Security Domain*, (CASD). C'est un type spécifique de SD. Il représente l'autorité de certification et son rôle est de faire appliquer la

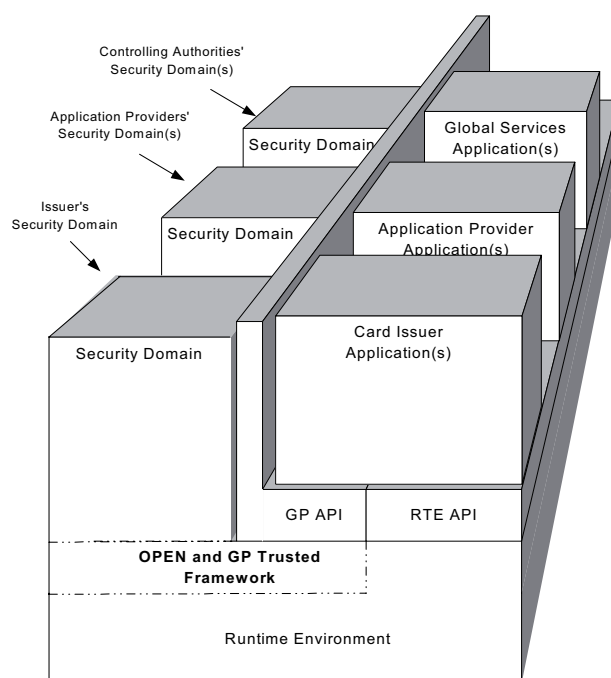


FIGURE 2.10 – Architecture d’une carte GlobalPlatform (source : GlobalPlatform)

politique de sécurité sur tout le contenu de la carte.

2.6.2.3.2 Les mécanismes de sécurité

Les mécanismes de sécurité qu’offre GlobalPlatform spécifient des méthodes pour :

- sécuriser les communications ;
- s’assurer que les applications chargées sont officiellement signées ;
- authentifier le porteur.

La sécurité des communications

Pour sécuriser les échanges entre la carte et une entité extérieure, GlobalPlatform spécifie deux mécanismes :

- l’authentification mutuelle,
- le canal sécurisé.

L’authentification mutuelle est la phase d’initiation d’un canal sécurisé pendant laquelle la carte et l’entité extérieure se donnent réciproquement l’assurance qu’elles communiquent avec une entité authentifiée *i.e.* qu’elles partagent le même secret. Quatre algorithmes et protocoles sont supportés : *Secure Channel Protocol* (SCP) 01 (déprécié) et 02 basés sur DES, SCP 03 basé sur AES et SCP 10 basé sur RSA.

Après que l'authentification mutuelle ait réussi, *le canal sécurisé* est initié. Durant la phase d'authentification mutuelle, le niveau de sécurité à appliquer à tous les messages suivant a été négocié. GlobalPlatform propose trois niveaux de sécurité :

- le niveau *authentification* correspond à un canal dans lequel les entités se sont authentifiées selon le mécanisme décrit précédemment ;
- le niveau *authentification et intégrité* assure que les messages reçus par une entité proviennent bien de l'autre entité, et que ni l'ordre des messages, ni leur contenu n'aient été altérés ;
- le niveau *authentification, intégrité et confidentialité* assure que les messages transmis entre les deux entités authentifiées ne sont pas visibles par une entité non authentifiée.

Note : le mécanisme d'authentification mutuelle décrit ci-dessus est le mécanisme *explicite* d'initiation du canal sécurisé. Il existe un mécanisme *implicite* qui consiste à initier un canal sécurisé à la réception de la première APDU protégée avec un code d'intégrité. Le niveau de sécurité est implicitement connu par la carte et l'entité extérieure ainsi que les clés à utiliser.

La signature des applications

Un fournisseur d'applications peut exiger de vérifier l'intégrité et l'authentification des applications qu'il charge sur la carte. GlobalPlatform propose un mécanisme répondant à ce besoin : le DAP (*Data Authentication Pattern*). Pour chaque morceau de code envoyé à la carte, une empreinte est calculée. Cette empreinte est ensuite signée et rajoutée à la structure du bloc envoyé. L'APSD dans lequel l'application est chargée doit posséder le privilège de vérification des DAP afin de fournir ce service au nom du fournisseur d'applications.

La vérification du porteur

GlobalPlatform, via le service global *Cardholder Verification Method* (CVM), fournit un mécanisme de vérification du porteur accessible à toutes les applications. Dans la version actuelle des spécifications⁵, seule la vérification du code PIN est supportée.

2.6.2.4 Le TEE

Un environnement d'exécution sécurisée (*Trusted Execution Environment*, TEE) est un environnement s'exécutant en parallèle du système d'exploitation afin de lui fournir des services de sécurité. Il existe plusieurs technologies pour implémenter

5. v2.2 (Mai 2009)

le TEE et le niveau de sécurité offert varie d'une implémentation à une autre. Le TEE hérite des travaux du *Trusted Computing Group* (TCG) sur le *Trusted Platform Module* (TPM) et le *Mobile Trusted Module* (MTM). Dans [60] et [61], Sarmenta, van Dijk *et al.* illustrent l'utilisation d'un TPM avec un système d'exploitation non sécurisé. Les TEE, selon leurs implémentations, peuvent offrir les services de sécurité suivants [62] :

- gestion de l'exécution des applications et de l'espace mémoire,
- gestion des fichiers et du stockage des données,
- communication inter-application,
- gestion des applications,
- communication avec le monde « extérieur »,
- gestion des périphériques,
- gestion des permissions et de l'identité de l'utilisateur.

Quelques implémentations du TEE dans l'industrie :

- ARM[®] TrustZone[®] Software
- GlobalPlatform GPD/STIP Platform
- Texas Instrument MShield[™]
- Trusted Logic Trusted Foundations[™] Software

La figure 2.11 illustre le concept de TEE tel qu'il est actuellement standardisé par le consortium GlobalPlatform [57, 63].

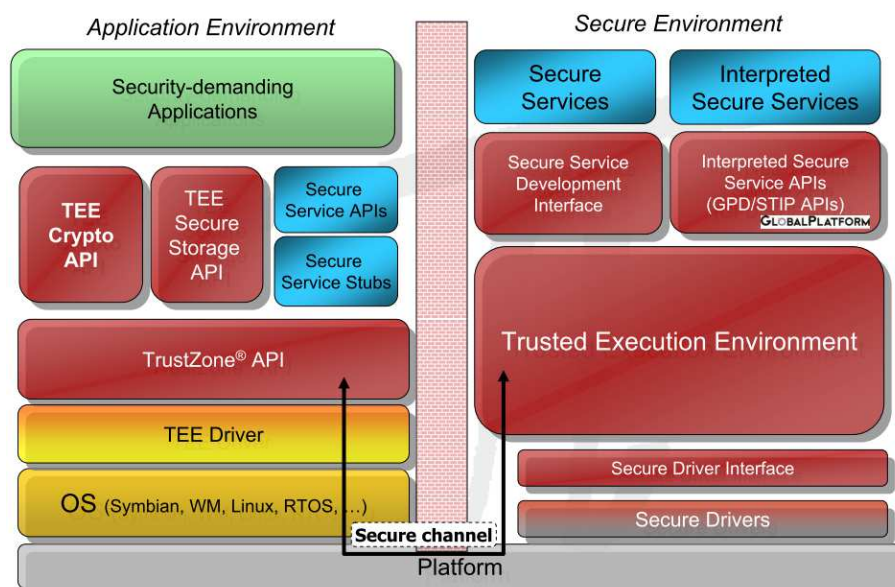


FIGURE 2.11 – Architecture du TEE en cours de standardisation à GlobalPlatform (source : Trusted Logic)

2.6.3 Comparaison et discussion

On peut séparer les quatre alternatives identifiées pour le SE (processeur du mobile, carte SIM (UICC), puce embarquée, carte mémoire amovible sécurisée) en deux familles : les solutions basées sur le système d'exploitation du mobile et les solutions basées sur une puce.

Dans [11], Reveilhac *et al.* estiment que l'implémentation du SE dans le processeur du mobile n'est viable ni à court ni à moyen terme. Le niveau de sécurité offert n'est pas suffisant et cette solution ne fait pas suffisamment l'objet d'une activité de standardisation. Cependant, l'avènement du TEE relance le débat. En effet, le TEE offre les propriétés attendues telles que définies à la section 2.6.1. Mais le manque de standardisation pour l'intégration du processeur du mobile dans l'architecture NFC reste un frein majeur. Les services mobiles sans contact s'appuient sur des applications dites *cardlets* ou *applets* mises en œuvre lors de l'émulation de carte. Le développement, la certification, l'installation et le déploiement de telles applications dans un TEE utilisent des méthodes propriétaires.

Pour les solutions basées sur une carte à puce, deux systèmes d'exploitation ouverts se démarquent : Java Card[™] et MULTOS[™]. Dans [64], Mayes *et al.* explique la raison du plus grand succès de Java Card[™] par rapport à MULTOS[™]. Le système d'exploitation MULTOS[™] a été conçu pour répondre aux plus hauts standards de sécurité et cela a pour effet d'avoir un cycle de développement très contrôlé, nécessitant plusieurs approbations et certifications avant de voir une application approuvée et chargée sur une carte. Ceci a dissuadé la majorité des développeurs qui se sont alors tournés vers le système Java Card[™]. Le niveau de sécurité offert par le système d'exploitation répond aux exigences du marché et les environnements de développements intégrés pour Java Card[™] sont nombreux.

Néanmoins, comme on l'a vu à la section 2.6.2.3, les systèmes d'exploitation ouverts cités ci-dessus ont des lacunes en termes de gestion de contenu applicatif et de cycle de vie. Ces lacunes ont été comblées par les spécifications du consortium GlobalPlatform rassemblant les acteurs majeurs de l'industrie de la carte à puce. Aujourd'hui, GlobalPlatform s'impose comme un standard de fait et est implémenté dans la quasi totalité des cartes à puce des domaines bancaires et télécommunications. Il est à noter que la plateforme GlobalPlatform a été portée pour les systèmes d'exploitation Java Card[™] et MULTOS[™].

Dans cette thèse, nous nous intéressons à un SE basé sur une carte à puce. Ce SE implémente la plateforme GlobalPlatform au-dessus d'une machine virtuelle Java Card[™].

2.7 La technologie NFC couplée à un SE GlobalPlatform

2.7.1 L'écosystème NFC

On peut définir l'écosystème NFC comme l'ensemble des acteurs qui contribuent à la mise au point, au développement et à la standardisation de la technologie NFC, ainsi qu'au déploiement et à l'utilisation par l'utilisateur final de services NFC. Il est généralement représenté de manière concentrique comme dans la Figure 2.12 parue dans les travaux de la GSMA⁶[15]. Les acteurs impliqués ainsi que leurs rôles respectifs sont détaillés ci-dessous [65, 15] :

L'utilisateur final utilise le mobile pour les communications et l'accès à des services mobiles. Il souscrit un contrat d'abonnement auprès d'un opérateur de télécommunications.

L'opérateur de télécommunications fournit l'ensemble des services mobiles à l'utilisateur (voix, SMS, emails, flux vidéos, ...).

Le fournisseur de service fournit un service mobile sans-contact à l'utilisateur (exemples de fournisseurs de service : banques, compagnies de transport public, émetteurs de programme de fidélité).

Le développeur d'application spécifie et développe les applications mobiles NFC (applications de paiement, fidélité, transport, ...).

Le TSM (Trusted Service Manager) distribue et gère de manière sécurisée les services des fournisseurs de service aux clients des opérateurs de télécommunications.

Les fabricants de carte SIM, de téléphones mobiles et de puces NFC conçoivent et produisent les composants matériels et logiciels nécessaires à la communication NFC et à l'exécution des services mobiles.

Les organismes de standardisation spécifient et implémentent un ensemble de standards pour le NFC pour assurer l'interopérabilité des applications et services NFC.

Les éditeurs de système (paiement, fidélité, ...) définissent et spécifient l'ensemble des fonctionnalités requises pour la mise en œuvre des services mobiles NFC.

6. GSMA : GSM Association. Association regroupant les plus grands opérateurs mondiaux de téléphonie mobile. Elle travaille depuis 2006 sur le NFC et a délivré beaucoup de documents de travail de référence

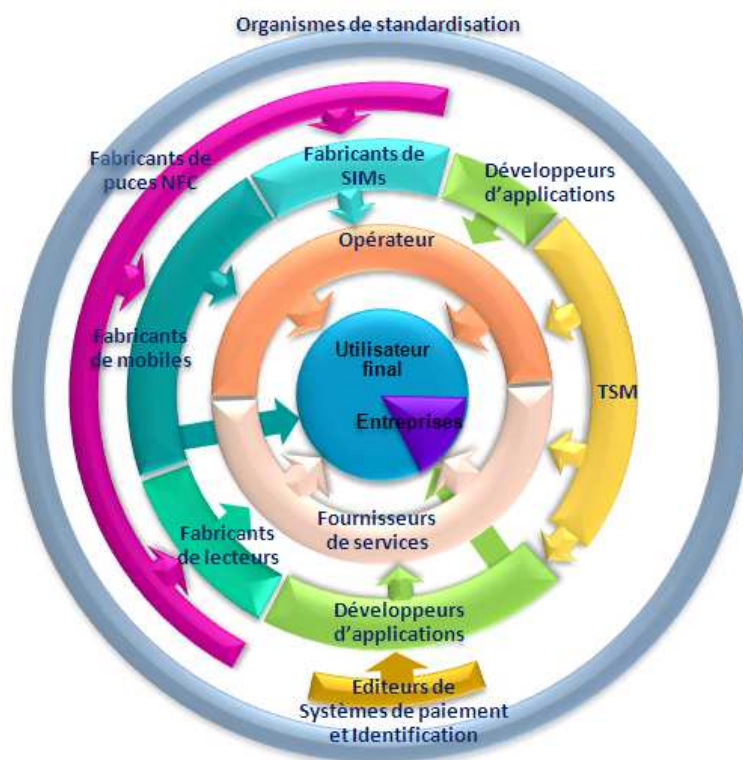


FIGURE 2.12 – Représentation de l'écosystème NFC (source : INSIDE Secure)

2.7.2 Le cycle de vie des services mobiles sans-contact

Le cycle de vie d'un service mobile sans contact est divisé en deux phases : la phase de déploiement sur le SE et la phase d'utilisation du service par l'utilisateur final. Les sections suivantes détaillent de manière fonctionnelle ces deux phases et nos contributions dans le cadre de cette thèse.

2.7.2.1 La phase de déploiement

La phase de déploiement d'un service mobile sans contact regroupe tous les cas d'utilisation liés à la gestion OTA tels que le déploiement, le verrouillage/déverrouillage, la mise à jour ou la suppression des applications constituant le service. Cette phase est gérée par un acteur ayant le rôle de TSM. Cet acteur peut être un opérateur de télécommunications ou un opérateur tiers. Les architectures mises en œuvre actuellement sont inspirées des mondes de la carte bancaire (statique) et de la carte SIM (modérément dynamique). Ces architectures montreront rapidement leurs limites lors de déploiements massifs sur des millions d'équipements potentiellement gérés par de multiples acteurs. Cette thèse s'attache dans un premier temps à définir une architecture d'aide au déploiement des services mobiles sans contact et à définir

une architecture permettant l'acceptation des paiements mobiles et sans contact avec un équipement NFC.

2.7.2.2 La phase d'utilisation du service

Lors de l'utilisation d'un service mobile sans contact, l'utilisateur final peut être amené à rencontrer les cas d'utilisation suivants : le changement de SE, le changement de mobile, le changement d'abonnement ou de numéro, perte ou vol du SE ou du mobile, ... Il est avant tout amené à présenter son mobile auprès d'un lecteur pour réaliser une transaction électronique. De par leur nature, ces transactions sans contact sont plus facilement sujettes à des attaques. Cette thèse s'attache dans un second temps à définir une plateforme d'analyse des transactions permettant d'implémenter de manière logicielle – et donc à bas coût – des attaques et permettant également de tester en amont pendant la phase de développement, les applications sensibles localisées dans le SE.

2.8 Conclusion

Dans ce chapitre, nous avons donné quelques définitions et exposé la problématique des services mobiles sans contact. Le contexte de cette thèse se situe dans la thématique des transactions sans contact, les applications sensibles comme le paiement et plus particulièrement le paiement mobile de proximité rendu possible par la technologie NFC et l'utilisation d'un SE GlobalPlatform. Nous intéressons au cycle de vie d'un service mobile sans contact et plus particulièrement à son déploiement et à son utilisation.

Les contributions de cette thèse sont de deux ordres. Dans un premier temps, nous définissons une plateforme d'aide au déploiement des services mobiles sans contact. Celle-ci s'appuie sur une modélisation à deux niveaux de la spécification GlobalPlatform. Nous définissons également un mécanisme de distribution des clés d'un SE à la demande permettant aux TSMs de se connecter puis de gérer au nom de son émetteur, un SE qui ne lui est pas connu. Puis, dans un second temps, nous nous attachons à mettre en œuvre une plateforme d'analyse des transactions de paiement. Cette plateforme, basée sur un outil existant développé au laboratoire GREYC, permet d'assurer la reproductibilité des attaques de l'état de l'art sur les transactions de paiement EMV, et ce de manière logicielle. Le chapitre suivant présente les contributions de cette thèse pour le déploiement des services mobiles sans contact et leur acceptation, notamment dans le domaine du paiement mobile de proximité.

Contributions au déploiement et à l'acceptation des services mobiles sans contact

Dans ce chapitre, nous adressons l'aide automatisée au déploiement des services mobiles sans contact ainsi que leur acceptation. Afin de répondre à ce problème, nous proposons une modélisation d'un SE. Cette modélisation est exploitée dans une plateforme de génération automatique de scripts de déploiement de services mobiles sans contact. Nous présentons également un mécanisme de distribution des clés du Secure Element à la demande et un ensemble d'architectures permettant l'acceptation des paiements mobile et sans contact avec un équipement NFC standard.

Sommaire

3.1	Introduction	36
3.2	Un mécanisme de distribution à la demande des clés des Secure Element	39
3.3	Proposition d'un terminal de paiement électronique sur téléphone mobile	45
3.4	Aide au déploiement de services mobiles sans contact	60
3.5	Conclusion	92

Présentation

QUELLE architecture pour le déploiement massif des services mobiles sans contact ? Quel modèle pour la gestion d'un SE par plusieurs TSMs ? Quelle architecture pour sécuriser le mode lecteur d'un équipement NFC ? Ce chapitre répond à ces questions en présentant un nouveau mécanisme de distribution des clés des SEs à la demande, une architecture sécurisée pour l'acceptation des paiements avec un mobile NFC et une plateforme de génération automatique de scripts de déploiement.

Mots clés :

Services mobiles, distribution des clés, secure element, mode lecteur NFC, acceptation de paiements mobiles, PCI DSS, PA-DSS, TSM, modélisation d'un SE, ontologie

Contributions de ce chapitre

- La définition d'un mécanisme de distribution des clés d'un SE à la demande. Certains éléments nécessaires à ce schéma sont en cours de standardisation au sein du consortium GlobalPlatform. Ce mécanisme suscite un grand intérêt auprès du consortium EMVCo.
- Trois architectures permettant de transformer un équipement NFC en terminal de paiement électronique sécurisé pour le paiement mobile et sans contact.
- La modélisation à deux dimensions d'un SE GlobalPlatform avec une ontologie. Cette représentation permet de modéliser à la fois les concepts statiques et dynamiques de la spécification GlobalPlatform.
- La mise en œuvre de l'ontologie dans une plateforme de génération automatique de scripts de déploiement de services mobiles sans contact.

3.1 Introduction

LE cycle de vie d'un service mobile est composé de quatre grandes phases comme l'illustre la figure 3.1 : la conception, le développement, le déploiement et l'utilisation par l'utilisateur final.

Les architectures de déploiement classiques s'appuient sur des systèmes de gestion de cartes statiques ou modérément dynamiques. Avec l'avènement de la technologie

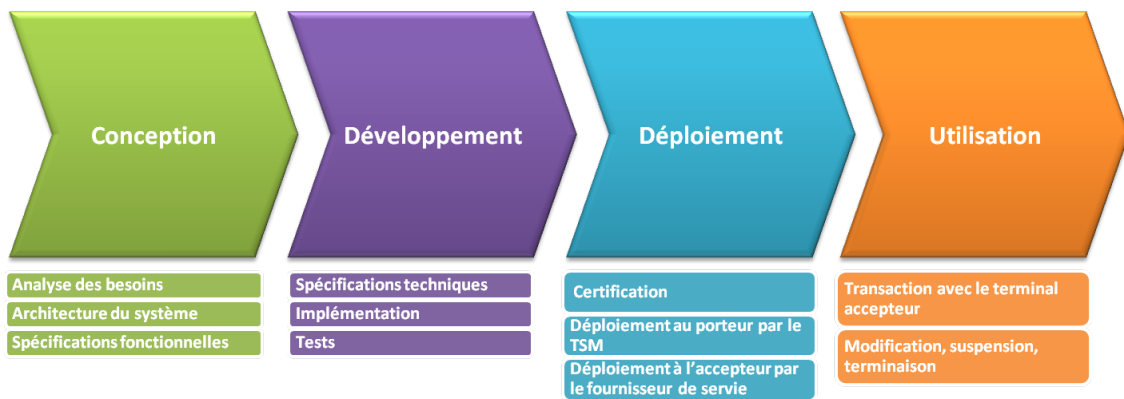
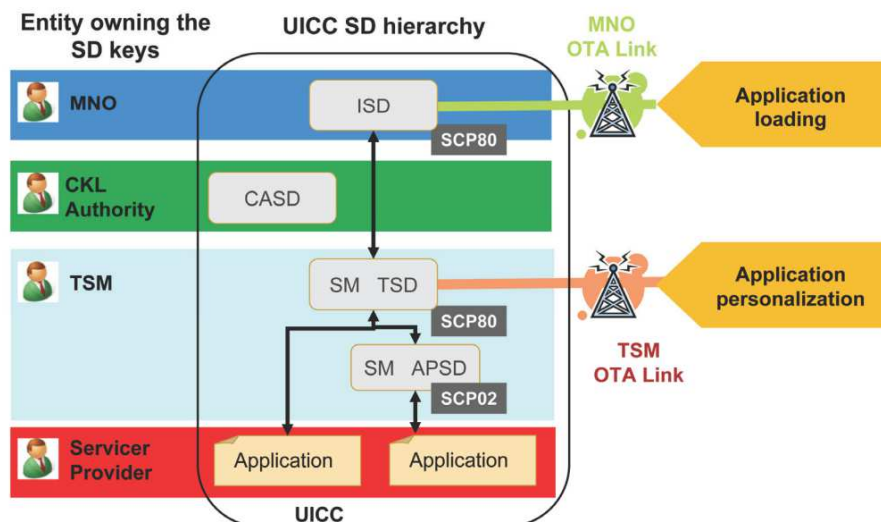


FIGURE 3.1 – Cycle de vie d'un service mobile sans contact

NFC, pratiquement tous les mobiles embarqueront un SE, environnement fortement dynamique pouvant être géré par plusieurs acteurs de l'écosystème et dans des modes de gestion différents comme l'illustrent les figures 3.2 et 3.3. Dans ce contexte, les architectures traditionnelles devront subir une forte modification pour s'adapter au déploiement des services mobiles sans contact. Une modélisation du SE dans un langage commun à tous les acteurs et permettant de représenter à la fois les aspects statiques et dynamiques de la gestion des SE pourrait palier aux faiblesses des systèmes traditionnels.

FIGURE 3.2 – Gestion en mode *Simple*, exemple de l'UICC (source : GlobalPlatform)

Le modèle d'émission des SE va également se complexifier avec leur intégration dans des équipements autres que les téléphones comme les ordinateurs, organisateurs personnels et autres tablettes. En effet, les fabricants de ces équipements n'ont

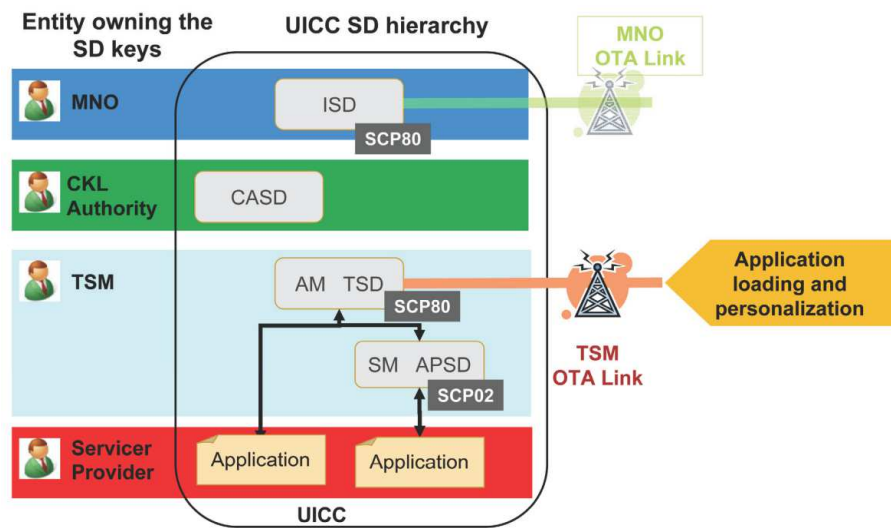


FIGURE 3.3 – Gestion en mode *Dual Management*, exemple de l'UICC (source : GlobalPlatform)

pas forcément vocation à intégrer des systèmes de gestion de clés ou à gérer des relations techniques ou commerciales avec les TSMs. Ainsi, un schéma de distribution des clés dans lequel le fabricant choisit l'entité qui gère ses clés (exemple : le fondateur du SE) et le « dialogue » avec les TSMs simplifierait l'adoption de la technologie NFC.

Le déploiement de services mobiles sans contact comprend également le déploiement des mêmes services sur les terminaux accepteurs. Dans le domaine du paiement, l'un des domaines les plus exigeants en terme de sécurité, l'acceptation des paiements sans contact et mobile implique l'acquisition d'un terminal spécifique ou la modification du terminal existant. Cette acquisition peut représenter un coût substantiel pour une catégorie de marchands ou ne pas être adaptée à l'activité de certains. L'implémentation d'une solution bas coût basée sur un équipement NFC contribuerait également à l'adoption du NFC.

Organisation du chapitre

Les sections suivantes présentent nos contributions. La section 3.2 présente une proposition de mécanisme de distribution des clés des SEs à la demande. La section 3.3 présente nos propositions d'architectures permettant à un équipement NFC d'accepter des paiements mobile et sans contact de manière sécurisée. La section 3.4 présente nos travaux sur la modélisation d'un SE GlobalPlatform et son application au déploiement des services mobiles sans contact.

3.2 Un mécanisme de distribution à la demande des clés des Secure Element

3.2.1 Introduction

Afin de pouvoir déployer un service mobile sans contact, le TSM chargé de cette opération doit posséder les clés du domaine de sécurité dans lequel il souhaite charger et installer ce service. Dans la majeure partie des cas, ces clés sont fournies au TSM par l'émetteur (l'entité qui émet l'équipement contenant le Secure Element) selon des accords pré-définis. Mais au cours de cette thèse réalisée en collaboration la société INSIDE Secure, nous avons rencontré des cas où cette hypothèse de départ ne pouvait être vérifiée, ceci pour diverses raisons : l'émetteur ne veut pas ou ne peut pas assurer l'opération d'un système de gestion des clés ou encore l'émetteur ne souhaite pas multiplier les accords et les échanges de clés avec les différents TSMs qui souhaiteraient s'interfacer avec ses équipements. Pour ces différentes raisons, nous avons été amenés à imaginer un système de distribution des clés qui émettrait les clés à la demande aux TSMs accrédités. Dans les sections suivantes, nous détaillons l'architecture et la cinématique de distribution des clés à la demande, ainsi que notre action afin de standardiser ce mécanisme auprès du consortium GlobalPlatform.

3.2.2 Description du problème

Étudions la cinématique du déploiement d'un service mobile sans contact.

1. Un utilisateur final émet une demande de souscription à un service mobile.
2. Le fournisseur d'applications délègue totalement la partie opérationnel du déploiement du service au TSM.

A ce stade, dans la majeure partie des cas, le TSM demande à l'émetteur du Secure Element les informations sur les capacités et le mode de gestion de ce Secure Element. Maintenant, considérons le cas suivant :

- Quelque soit le mode de management du Secure Element, le TSM a le privilège associé.
- Pour des raisons techniques et/ou commerciales, l'émetteur sous-traite la gestion des clés à un tiers de confiance.

Ainsi, les étapes suivantes du déploiement du service mobile sont :

3. Le TSM identifie le tiers de confiance détenant les clés du Secure Element.
4. Le TSM demande les clés du Secure Element au tiers de confiance.

5. Le tiers de confiance génère les clés du Secure Element à partir de la clé maître et des données d'identification fournies par le TSM.
6. Les clés sont envoyées au TSM.
7. Le TSM ouvre une session sécurisée avec le Secure Element.
8. Le TSM effectue une rotation de clés, i.e. il « injecte » son propre jeu de clés dans le Secure Element.
9. Le TSM procède au déploiement du service mobile en envoyant un script de commandes au travers du canal sécurisé précédemment ouvert.

Cependant, un obstacle se dresse dans la réalisation de ce scénario. En effet, on retrouve dans le Secure Element des données permettant d'identifier seulement :

- le fabricant de la puce et du système d'exploitation via les données de production du Secure Element (*Card Production Life Cycle, CPLC*),
- l'émetteur via son numéro d'identification (*Issuer Identification Number, IIN*).

Le SE ne contient pas d'information permettant d'identifier le tiers de confiance, le TSM ne peut donc pas procéder à l'étape 3. Nous proposons dans les sections suivantes une méthode pour permettre l'identification du tiers de confiance et une architecture de distribution des clés.

3.2.3 Propositions pour l'identification du tiers de confiance

Ainsi, dans les spécifications GlobalPlatform, aucun mécanisme n'est prévu pour identifier une autre entité que les fabricants et l'émetteur. Pour palier ce problème, nous avons imaginé deux solutions décrites dans les sections suivantes.

3.2.3.1 Solution n°1 : Étendre les données de reconnaissance de la carte

Dans cette solution, nous proposons d'étendre les données de reconnaissance de la carte (*Card Recognition Data, CRD*, tag '66') pour y stocker un identifiant permettant de pointer sur le tiers de confiance. Le format des CRD est donné à la figure 3.4. Comme les CRD sont limités à 127 octets, cet identifiant doit être court (quelques octets). Ceci présume donc de l'existence d'un « annuaire » central (électronique sous la forme d'un serveur ou non-électronique sous la forme d'une spécification) faisant le lien entre l'identifiant court et l'adresse des serveurs du tiers de confiance.

Tag	Explanation	Length	Value	Presence
'66'	Tag for 'Card Data'	variable - see note 1	Data objects identified in 7816-6, including tag '73'	Mandatory
'73'	Tag for 'Card Recognition Data'	variable	Data objects listed below	Mandatory
'06'	Universal tag for "Object Identifier" (OID)	variable	{globalPlatform 1} OID for Card Recognition Data, also identifies Global Platform as the Tag Allocation Authority	Mandatory
'60' '06'	Application tag 0 'OID' tag	variable variable	{globalPlatform 2 v} OID for Card Management Type and Version - see note 2	Mandatory
'63' '06'	Application tag 3 'OID' tag	variable variable	{globalPlatform 3} OID for Card Identification Scheme - see note 3	Mandatory
'64' '06'	Application tag 4 'OID' tag	variable variable	{globalPlatform 4 scp i} OID for Secure Channel Protocol of the Issuer Security Domain and its implementation options- see note 4	Mandatory
'65'	Application tag 5	variable	Card configuration details - see note 5	Optional
'66'	Application tag 6	variable	Card / chip details - see note 6	Optional
'67'	Application tag 7	variable	Issuer Security Domain's Trust Point certificate information - see note 7	Optional
'68'	Application tag 8	variable	Issuer Security Domain certificate information - see note 8	Conditional

FIGURE 3.4 – Structure des données de reconnaissance de la carte (Source : Global-Platform)

3.2.3.2 Solution n°2 : Ajouter un tag dans les données du domaine de sécurité

Dans cette solution, nous proposons d'ajouter un tag (*i.e.* un conteneur de données) dans les données du domaine de sécurité. Ce tag contiendrait une URL pointant directement sur les serveurs du tiers de confiance. L'URL pourrait être de la forme `http://service.montsm.net/secureelement-management/v1/index.htm`. Le stockage d'une URL est préférable au stockage d'un couple adresse IP/port TCP car il est plus flexible. En effet, l'URL fait appel à des services DNS et rend donc transparente une éventuelle modification de la topologie réseau du tiers de confiance.

3.2.3.3 Étude comparative

Le tableau 3.1 synthétise les avantages et inconvénients de chacune des solutions.

Il apparaît dans le tableau 3.1 que la meilleure solution consiste à ajouter un tag dans les données du domaine de sécurité. Elle permet de stocker une URL entière et ainsi un lien direct vers le tiers de confiance. L'URL est accessible par le biais d'une commande GET DATA ne nécessitant pas l'établissement d'un canal sécurisé. Dans les sections suivantes, nous décrivons l'architecture de la solution proposée

	Avantages	Inconvénients
Card Recognition Data	Stockage d'une donnée courte Lu avec un GET DATA (non sécurisé)	Limité en taille Nécessite un annuaire
Tag	Pointeur direct Non limité en taille Lu avec un GET DATA (non sécurisé)	–

TABLE 3.1: Comparaison des solutions proposées

ainsi que la cinématique de distribution à la demande des clés du Secure Element en utilisant cet élément de données.

3.2.4 Architecture proposée

3.2.4.1 Serveur de clés

Le serveur mis en œuvre par le tiers de confiance expose un Web Service qui publie les fonctions accessibles aux différents TSMs au format WSDL (*Web Service Definition Language*). La connexion et l'authentification entre le TSM et le tiers de confiance sont réalisées lors de l'établissement d'un VPN (*Virtual Private Network*) basé sur le protocole SSL (*Security Socket Layer*) ou son successeur TLS (*Transport Layer Security*) en mode authentification du client comme décrit dans [66]. Les messages échangés respectent le format XML (*eXtended Markup Language*) et le protocole SOAP (*Simple Object Access Protocol*).

3.2.4.2 Passerelle vers le Secure Element

Dans [67], GlobalPlatform définit un mécanisme d'administration à distance d'un Secure Element en utilisant le protocole HTTP. En particulier, cette spécification décrit une application dite « *Administration Agent* » et dont le rôle est de désencapsuler les commandes APDU reçues du serveur dans une enveloppe HTTP et de les router au Secure Element.

3.2.5 Cinématique

La cinématique de la distribution à la demande des clés du Secure Element est représentée à la figure 3.5. Les différentes étapes sont détaillées ci-dessous.

Pré-requis : Le TSM et le tiers de confiance doivent échanger une clé de transport *KEK* (*Key Exchange Key*).

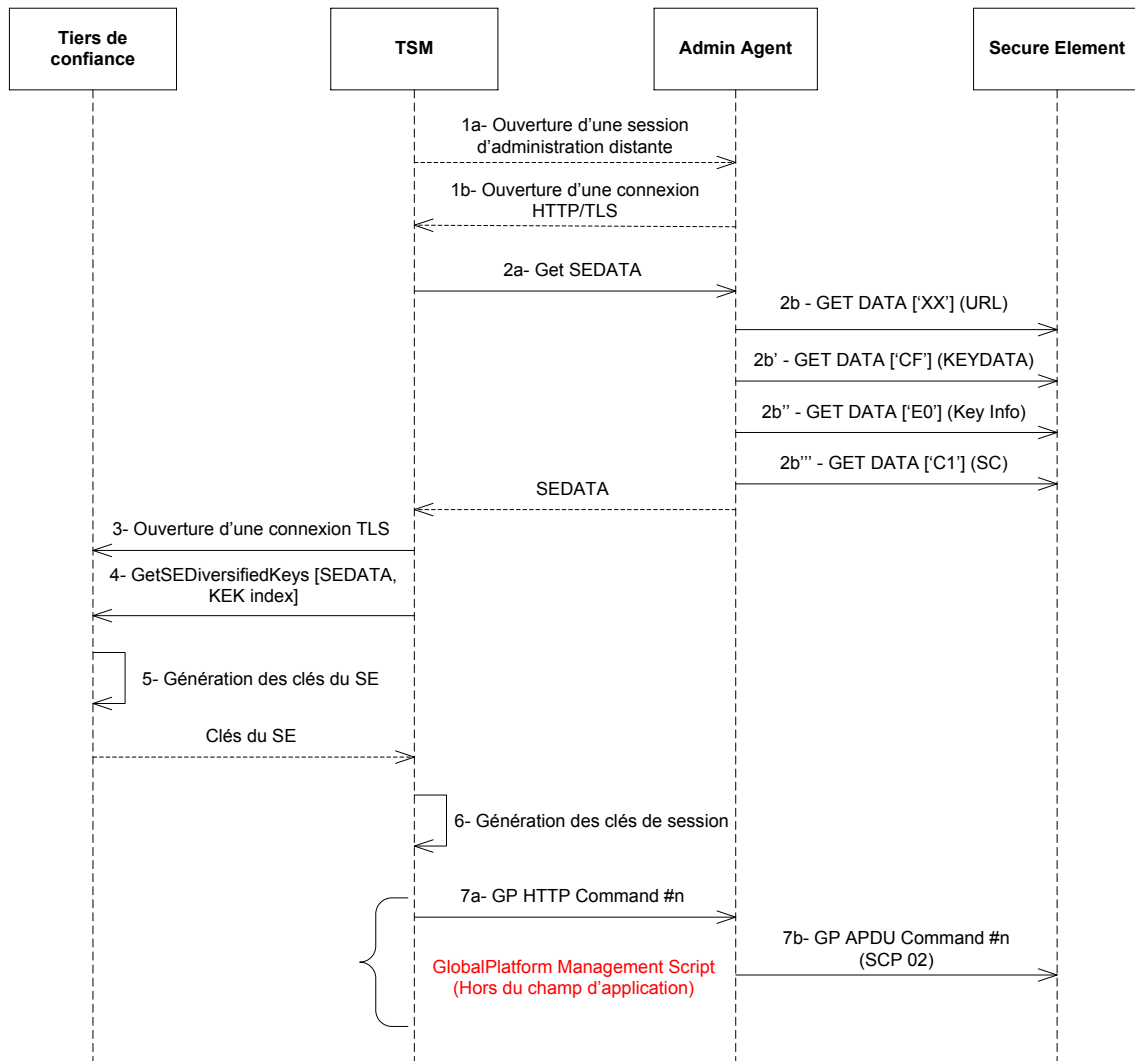


FIGURE 3.5 – Cinématique de la distribution des clés à la demande

Étape 1 : Le TSM déclenche une session d’administration à distance. L’Admin Agent ouvre alors une connexion TLS avec le TSM tel que défini dans [67].

Étape 2 : Le TSM demande à l’Admin Agent de lui fournir les informations relatives au Secure Element suivantes :

- l’URL du tiers de confiance (tag à définir),
- les données d’identification du Secure Element (KEYDATA, tag 'CF'),
- les informations permettant d’identifier le jeu de clés utilisé (tag 'E0'),
- le compteur de session sécurisée maintenu par le domaine de sécurité (*Sequence Counter*, tag 'C1'). Ce compteur est utilisé à l’étape 6 pour le calcul des clés de session.

Étape 3 : En utilisant l’URL, le TSM ouvre une connexion SSL/TLS avec le tiers

de confiance.

Étape 4 : Le TSM invoque la méthode `WSGetDiversifiedKeys` avec les informations du Secure Element et l'index de la *KEK* utilisée comme paramètres.

Étape 5 : Le tiers de confiance génère les clés diversifiées du Secure Element (K_{ENC} , K_{MAC} , K_{DEK}) et les retourne au TSM.

Étape 6 : A partir des clés diversifiées et du compteur de session, le TSM génère les clés de session et ouvre une session sécurisée avec le Secure Element.

Étape 7 : Le TSM génère les scripts de gestion, les encapsule dans une enveloppe HTTP et les envoie à l'Admin Agent. L'Admin Agent désencapsule les APDUs et les route au Secure Element (hors du champ d'application).

Dans la section suivante, nous expliquons notre action auprès des consortiums de standardisation GlobalPlatform et EMVCo afin de voir ce nouveau schéma de distribution des clés adopté par l'ensemble des acteurs de l'écosystème NFC.

3.2.6 Validation

Nous pensons que ce mécanisme de distribution est un élément clé pour la large adoption de la technologie NFC. Les premiers terminaux dans lesquels le NFC a été intégré sont les téléphones mobiles où les différents acteurs sont plus ou moins habitués à l'exercice de la gestion des clés. Mais si on considère que n'importe quel terminal peut être équipé de la technologie NFC, on se rend alors compte que des manufacturiers d'équipements tels que les ordinateurs portables ou les tablettes devront alors s'y confronter ainsi qu'à l'échange de ces clés avec les TSMs. Notre solution permet alors à ces acteurs de s'affranchir de cette gestion qui leur est parfois inconnue.

Afin de voir notre solution adoptée par les acteurs de l'écosystème NFC, nous avons souhaité le faire valider par le consortium GlobalPlatform. La présentation de notre schéma a eu lieu lors d'une réunion de la *GlobalPlatform Mobile Task Force* [68], à laquelle nous sommes régulièrement conviés en tant que membre. Elle fut assez bien reçue et il a été décidé de transmettre les points d'actions aux deux comités concernés :

- la *Systems Committee* pour la cinématique et les échanges entre le TSM et le tiers de confiance,
- et le *Card Committee* pour l'ajout du tag dans les données du domaine de sécurité.

Après plusieurs réunions et analyses des spécifications, il a été conclu que :

- les spécifications système, en particulier [58], contiennent déjà tous les éléments permettant de réaliser notre solution,
- un tag pouvant contenir l'URL d'un tiers de confiance sera ajouté dans la prochaine version de maintenance des spécifications. Nous avons soumis une proposition qui sera intégrée à l'amendement C de la spécification carte version 2.2.1.

De plus, parmi les membres de GlobalPlatform certains sont également membres du consortium EMVCo dont la mission est de définir et maintenir les spécifications de paiement par carte à puce EMV (Eurocard, Mastercard, Visa). EMVCo est en train de définir un profil précisant les caractéristiques supportées et non supportées par EMVCo pour la certification d'applets de paiement sur un Secure Element embarqué. Dans ce cadre, EMVCo requiert que les clés des domaines de sécurité hébergeant les applications de paiement soient échangées entre un TSM certifié et l'entité certifiée qui les a chargées. En d'autres termes, dans le cadre du paiement EMV, l'émetteur de l'équipement NFC équipé d'un Secure Element embarqué, n'est plus un maillon dans la chaîne de gestion des clés. C'est pourquoi notre proposition de système de distribution des clés mettant en relation un TSM et un tiers de confiance certifié a reçu un écho positif de EMVCo.

3.2.7 Discussion

La solution de distribution des clés d'un Secure Element à la demande répond à un réel besoin et s'avère être un élément majeur pour l'adoption massive de la technologie NFC par les manufacturiers d'équipements électroniques. Elle a été présentée aux organismes de standardisation du domaine et y a reçu des échos positifs. GlobalPlatform ajoutera un nouveau tag dans la prochaine version de sa spécification carte et EMVCo y voit un moyen de rendre possible la chaîne de gestion des clés définie dans son profil pour un Secure Element embarqué.

Dans la section suivante, nous proposons trois architectures sécurisées pour transformer un appareil mobile doté de la technologie NFC en terminal de paiement électronique sans contact.

3.3 Proposition d'un terminal de paiement électronique sur téléphone mobile

Dans cette section, nous proposons un ensemble d'architectures permettant d'accepter des transactions de paiement sans contact et/ou mobile avec un téléphone.

3.3.1 Introduction

L'utilisation du mobile, qui devient de plus en plus indispensable dans la vie de tous les jours, permet d'implémenter un terminal de paiement mobile à bas coût et adapté à une certaine catégorie de marchands. Des solutions existent déjà pour permettre ce type de transactions, mais la majeure partie utilisent les données de la piste ou celles embossées et imprimées au recto et au verso de la carte. L'intégration de la technologie NFC dans un mobile et l'utilisation de son mode lecteur permet de communiquer avec une carte sans contact ou un mobile et d'initier une transaction de paiement. Cependant, cette transaction doit être exécutée dans des conditions optimales de sécurité et de respect de la vie privée de l'utilisateur.

Après avoir énoncé les propriétés attendues d'un tel équipement, nous faisons un tour d'horizon des standards en vigueur pour le paiement et la sécurité de leur acceptation puis exposons notre contribution.

3.3.2 Propriétés attendues

Les propriétés attendues d'un équipement mobile NFC capable d'accepter des transactions de paiement sans contact et mobiles sont les suivantes :

sécurisé, l'équipement doit offrir un niveau de sécurité satisfaisant du point de vue du porteur (le client), de l'accepteur (le marchand), de l'acquéreur (la banque du marchand) et de l'émetteur (la banque du client)

certifiable selon les standards en vigueur, la solution doit être conforme aux standards de sécurité en vigueur ayant trait à l'acceptation des paiements par carte à puce sans contact (*i.e.* PCI, EMVCo, ...)

conforme aux standards de paiement, la solution doit intégrer des applications de paiement qui respectent les spécifications en vigueur tant au niveau communication en radio fréquence, cinématique de paiement, temps de transaction, ... (*i.e.* EMVCo niveau 1, Visa, MasterCard, ...)

3.3.3 Les certifications

Cette section décrit les certifications en vigueur pour les solutions de terminal de paiement, *i.e.* les certifications PCI (*Payment Card Industry*) et EMVCo.

3.3.3.1 PCI DSS

PCI-DSS (*Payment Card Industry Data Security Standard*) [69] est un standard de sécurité défini par le *Payment Card Industry Security Standards Council* (PCI

SSC). Il a été créé pour aider les organismes qui traitent des transactions de paiement à lutter contre la fraude grâce à des contrôles de sécurité accrus et en s'assurant de la non exposition des données sensibles.

L'évaluation de sécurité de la norme PCI DSS s'appuie sur les 12 clauses suivantes :

- Création et gestion d'un réseau sécurisé
 1. Installer et gérer une configuration de pare-feu pour protéger les données des titulaires de cartes.
 2. Ne pas utiliser les mots de passe système et autres paramètres de sécurité par défaut définis par le fournisseur.
- Protection des données des titulaires de cartes de crédit
 3. Protéger les données des titulaires de cartes stockées.
 4. Chiffrer la transmission des données des titulaires de cartes sur les réseaux publics ouverts.
- Mise à jour d'un programme de gestion des vulnérabilités
 5. Utiliser des logiciels antivirus et les mettre à jour régulièrement.
 6. Développer et gérer des systèmes et des applications sécurisés.
- Mise en œuvre de mesures de contrôle d'accès strictes
 7. Restreindre l'accès aux données des titulaires de cartes aux seuls individus qui doivent les connaître.
 8. Affecter un ID unique à chaque utilisateur d'ordinateur.
 9. Restreindre l'accès physique aux données des titulaires de cartes.
- Surveillance et test réguliers des réseaux
 10. Effectuer le suivi et surveiller tous les accès aux ressources réseau et aux données des titulaires de cartes.
 11. Tester régulièrement les processus et les systèmes de sécurité.
- Gestion d'une politique de sécurité des informations
 12. Gérer une politique qui assure la sécurité des informations des employés et des sous-traitants.

3.3.3.2 PCI PTS

PCI PTS (*Payment Card Industry PIN Transaction Security*) définit la sécurité pour l'entrée du PIN, sa vérification hors ligne ainsi que la transmission en ligne des données PIN. Les exigences sont présentées dans quatre manuels détaillant les modules suivants [70] :

Encrypting PIN Pads (EPP) : définit la sécurité PIN pour les terminaux et les DAB (Distributeur Automatique de Billets)

Point of Sale PIN Entry Devices (POS-PED) : définit un large éventail d'exigences concernant la sécurité logique et physique au POS

Unattended Payment Terminals (UPT) : décrit les exigences de sécurité applicables aux terminaux autonomes (à l'exception de distributeurs automatiques de billets) qui permettent l'entrée PIN

Hardware Security Modules (HSM) : définit les exigences de sécurité pour la conception logique et physique d'HSM ainsi que pour leur processus de fabrication et d'installation

3.3.3.3 PA-DSS

La norme PA-DSS (*Payment Application Data Security Standard*) est un standard de sécurité global créé par le PCI SSC. Le programme était précédemment supervisé par Visa et portait le nom de *Payment Application Best Practices* (PABP). Le but de PA-DSS est de faciliter le développement d'applications de paiement sécurisées qui ne gardent pas en mémoire de données dites "interdites" comme le contenu complet de la piste magnétique, CVV2¹ ou des données relatives au PIN ; et de s'assurer que ces applications sont conformes à PCI DSS. Pour être conforme, une application doit répondre aux 14 critères suivants :

1. Ne pas conserver la totalité des données de bande magnétique, de code ou de valeur de validation de carte (CAV2, CID, CVC2, CVV2), ou de bloc PIN
2. Protéger les données de titulaire de carte stockées.
3. Fournir des fonctions d'authentification sécurisées.
4. Enregistrer l'activité des applications de paiement.
5. Développer des applications de paiement sécurisé.
6. Protéger les transmissions sans fil.
7. Tester les applications de paiement pour gérer les vulnérabilités.
8. Permettre la mise en œuvre de réseaux sécurisés.
9. Les données de titulaire de carte ne doivent jamais être stockées sur un serveur connecté à Internet.
10. Permettre des mises à jour logicielles à distance sécurisées.
11. Permettre un accès à distance sécurisé à l'application de paiement.
12. Chiffrer le trafic sensible transitant par les réseaux publics.

1. Le CVV2 (Card Verification Value), ou numéro de sécurité, est une caractéristique de sécurité importante qui permet de limiter les fraudes dans le contexte de transactions réalisées en ligne ou par téléphone.

13. Chiffrer tous les accès administratifs non-console.
14. Gérer la documentation fournissant des instructions et les programmes de formation pour les clients, les revendeurs et les intégrateurs.

3.3.3.4 PCI et le paiement mobile

La dernière version des spécifications PCI (PCI DSS v2) a été publiée en octobre 2010 pour une mise en application en janvier 2011. Le paiement mobile étant une technologie très récente, il était trop tôt pour que cette version de spécifications n'apporte un schéma clair pour la certification des terminaux de paiement mobile. Au contraire, le comité des standards de sécurité PCI a fait part dans un communiqué du 24 juin 2011 [71] de sa volonté à s'accorder une période supplémentaire d'évaluation de l'écosystème de l'acceptation des paiements mobiles. Cependant, ce communiqué apporte quelques éclaircissements afin de guider au mieux vers la conformité les éditeurs pendant cette période transitoire. Le comité a donc séparé les solutions d'acceptation de paiement mobile en 3 catégories :

- Catégorie 1 : l'application de paiement opère sur terminal mobile approuvé PCI PTS
- Catégorie 2 : l'application de paiement remplit les critères suivants :
 - l'application de paiement est fournie en tant que solution intégrée dans un terminal mobile spécifique
 - le terminal mobile a été conçu pour accomplir la fonction unique d'acceptation de paiement mobile
 - l'application de paiement fournit un environnement permettant au marchand de maintenir sa conformité au standard PCI DSS
- Catégorie 3 : l'application de paiement opère sur un équipement mobile qui n'est pas spécifiquement dédié à la fonction de paiement

Le comité ajoute à cette catégorisation des terminaux mobiles l'applicabilité de la certification PA-DSS. Les terminaux des catégories 1 et 2 sont éligibles à la certification PA-DSS, alors qu'il n'y a pas encore de certification applicable aux terminaux de la 3ème catégorie. Cependant, il est conseillé de respecter les recommandations PA-DSS.

Il apparaît à la lumière de ces éclaircissements que la solution qui nous intéresse dans ce chapitre tombe dans la catégorie 3. La certification PA-DSS n'est donc pas une obligation mais PCI recommande tout de même de suivre les recommandations PA-DSS avant de se rapprocher le plus possible de ce que sera la prochaine certification pour terminaux mobiles.

3.3.3.5 EMV

Une certification EMV est sous la responsabilité d'EMVCo, société de droit américain représentant MasterCard International, Visa International, JCB International et American Express. EMVCo a écrit et publié des spécifications et des cas de test pour les terminaux de paiement, qui sont à la base d'une certification EMV. Ces spécifications couvrent les applications de paiement (EMV niveau 2) comme les interfaces hardware (EMV niveau 1).

3.3.3.5.1 EMV niveau 1

La certification EMV niveau 1 s'assure de la conformité des terminaux avec les spécifications EMV [54] Livre 1 Partie 1. Elle introduit la notion de *Interface Module (IFM)* qui est composé du lecteur de carte à puce et des composantes matérielles et logicielles nécessaires à l'alimentation de la puce et à la communication. La figure 3.6 montre un modèle d'IFM contenu dans un terminal et les périmètres de tests qui sont définis autour. Seuls les périmètres A et B font l'objet de la certification niveau 1.

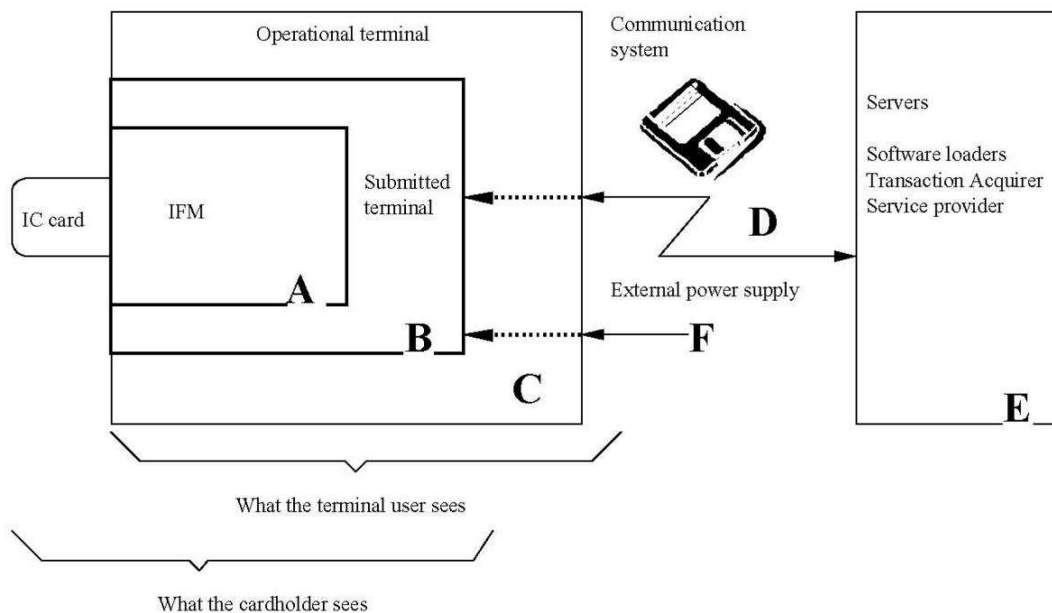


FIGURE 3.6 – Modèle d'IFM et de terminal (source : EMVCo)

3.3.3.5.2 EMV niveau 2

La certification EMV niveau 2 s'assure de la conformité des terminaux avec les spécifications EMV [54] Livre 1 - Partie 2, Livre 2 et Livre 4.

3.3.4 État de l'art des terminaux de paiement sans contact

Dans cette section, nous présentons les travaux de l'état de l'art dans le domaine du paiement sans contact et mobile, le paiement sans contact et mobile étant basé sur la même technologie. Nous présentons d'abord les travaux dans le domaine académique puis dans l'industrie. Nous montrons que les travaux académiques sur ce sujet se font rare et discutons les différentes contributions.

3.3.4.1 Les terminaux de paiement sans contact dans la littérature scientifique

Les terminaux sans contact ne sont pas un sujet très couvert dans la littérature. Dans une récente revue de la littérature (2008), Dalhberg *et al.* [72] pointent du doigt le fait que la plupart des papiers publiés sur le sujet couvrent des points techniques (exemple : sécurité, protocole) et des études orientées vers le consommateur (exemple : adoption de la technologie). De plus, comme le font remarqué Ondrus *et al.* dans [73], il n'existe que peu de papiers s'intéressant au potentiel de la technologie NFC et ces papiers sont très généralistes (exemples [74, 34]). Ceci peut être expliqué par la récente émergence de cette technologie et donc de la recherche à son sujet. De plus, les terminaux sont considérés comme des équipements hautement sécurisés situés chez les marchands et contrôlé exclusivement par les banques. Parmi les quelques papiers abordant ce sujet, on peut citer [75] dans lequel un téléphone mobile est transformé en un assistant à la vente qui affiche les informations sur un produit en lisant son étiquette RFID. Cependant, le scénario qui nous intéresse, *i.e.* le paiement de proximité, nécessite une architecture plus robuste car elle manipule des données sensibles.

3.3.4.2 Les terminaux de paiement sans contact dans l'industrie

Dans l'industrie, on trouve les terminaux de paiement sans contact sous diverses formes. On peut les trouver "sans surveillance", sous forme de lecteurs sans contact connecté à un terminal existant ou comme terminal sans contact intégré. Les terminaux sans surveillance sont situés sur les automates à carburant, péages ou autres distributeurs. Ils améliorent la vitesse de la transaction et la qualité de l'expérience utilisateur. Les lecteurs sans contact connectés à un terminal ou à n points de vente gèrent la partie radio-fréquence et la transaction de paiement. A la fin de la transaction, il envoie au terminal connecté les données à envoyer à la banque acquéreur. Parmi ces produits, on peut citer les terminaux ViVOPay[™] 4500 et 5000 du constructeur ViVOtech[©]. Les terminaux intégrés sont la nouvelle génération de terminaux

sans contact. Ils embarquent dans le même équipement tout le matériel et logiciel pour mener une transaction de paiement sans contact (antenne, micro-contrôleur, applications de paiement ...).



FIGURE 3.7 – Exemples de lecteurs et terminaux sans contact

3.3.4.3 Discussion

Les solutions présentées ont des caractéristiques communes telles que la facilité, la vitesse et la commodité de la transaction. La plupart d'entre elles sont destinées à être posées sur le comptoir du marchand. Très peu de ces solutions sont destinées à être mobiles, pour les commerçants itinérants. Cela peut s'expliquer par la nécessité (selon les pays) d'être connecté à la banque acquéreur et des coûts supplémentaires de télécommunications. Un point mérite d'être soulevé, celui du manque de terminaux sur le marché capables d'accepter les paiements mobiles NFC, i.e. lorsque la carte bancaire est émulée par un mobile. Ces paiements diffèrent des paiements standards sans contact du fait que, dans la majorité des cas, l'accès à la fonctionnalité de paiement est protégé par un code personnel. De ce fait, une interaction avec l'utilisateur est nécessaire afin de l'informer que la transaction en cours ne pourra aboutir que lorsque celui-ci aura saisi son code sur le mobile. L'autre différence majeure avec le paiement sans contact est la remise à zéro des compteurs *offline*. Les compteurs *offline* sont utilisés comme procédure de gestion du risque. Il en existe deux principaux : le compteur du nombre de transactions consécutives réalisées *offline* et le montant cumulatif des transactions réalisées *offline*, i.e. sans se connecter à la banque émettrice. Lorsque ces compteurs atteignent leurs limites, ils nécessitent d'être remis à zéro. Ceci est parfaitement géré avec les cartes *dual-interface* car elles sont également équipées d'une puce contact qui permet de réaliser cette procédure de manière sécurisée. Dans le paiement mobile, le passage de l'interface sans contact à l'interface contact n'est pas possible et de nouvelles procédures doivent donc être définies.

3.3.5 Un terminal de paiement électronique sur un téléphone NFC

Nous présentons dans cette section, trois propositions d'architecture de terminal de paiement électroniques basées sur la technologie NFC embarquée dans un équipement mobile. Nous considérons dans ces architectures les trois composants suivants : le contrôleur NFC , le Secure Element et un module à accès sécurisé (SAM) qui héberge des clés cryptographiques.

3.3.5.1 Tous les composants dans un module externe

Cette architecture est illustrée à la figure 3.8. Dans cette architecture, le contrôleur NFC et le Secure Element sont embarqués dans un module externe connecté sur un port externe du mobile. Le Secure Element accueille les applications de paiement, un SAM logiciel et des bibliothèques cryptographiques dédiées. Le module est connecté au processeur du mobile via un lien physique qui permet de le piloter pour déclencher une transaction de paiement. L'interface utilisateur est exécutée depuis le processeur d'applications.

Lorsque le commerçant veut procéder à une transaction de paiement, une commande est envoyée au module externe afin qu'il rentre en phase de « *polling* », c'est-à-dire une boucle de détection d'une carte dans le champ émis par le module. Lorsqu'une carte est détectée, l'application de paiement correspondante est invoquée et effectue la transaction. L'IHM est alors utilisée pour afficher les détails de la transaction et éventuellement capturer la signature du client grâce à un écran tactile.

3.3.5.2 Les applications de paiement dans le processeur d'application et la cryptographie dans le Secure Element

Dans cette architecture, illustrée à la figure 3.9, les applications de paiement sont localisées dans le processeur d'applications (PA) du mobile. Le SAM ainsi que les bibliothèques cryptographiques sont hébergées de manière sécurisée dans le Secure Element. Le SAM contient les clés publiques permettant au terminal de valider les certificats de la carte et éventuellement des clés symétriques pour sécuriser les communications avec le fournisseur de service de paiement (exemple : un banque) au travers du réseau de l'opérateur. Une API permet aux applications de paiement de faire appel aux bibliothèques cryptographiques dans le Secure Element. Cette API et les applications de paiement peuvent être sécurisées en s'appuyant sur un environnement d'exécution (*Trusted Execution Environment*, TEE) sécurisé tel que défini par GlobalPlatform dans [57]. L'objectif principal de la TEE est d'offrir une plateforme basée sur deux « mondes » :

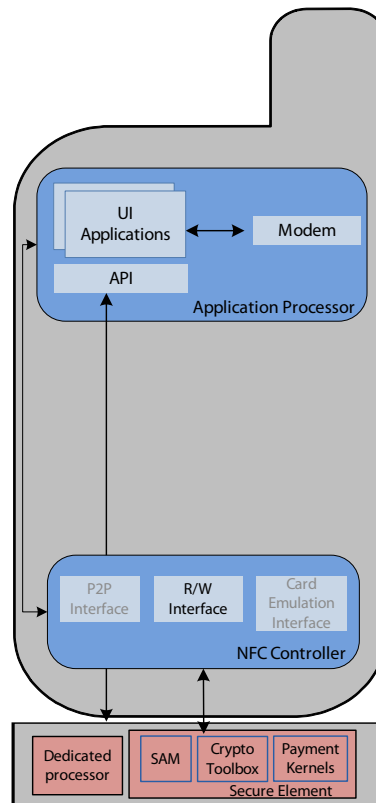


FIGURE 3.8 – Les applications de paiement et le SAM dans un module externe

un monde sécurisé dans lequel l'exécution de code, l'accès aux entrées/sorties et aux périphériques tels que le clavier et l'écran sont sécurisés ; et le monde « normal » dans lequel les applications sont exécutées de manière standard. Ces dernières peuvent accéder à des applications sécurisées appelées *services* au travers d'un canal sécurisé entre les deux mondes.

Lorsqu'une carte est détectée dans le champ radio-fréquence du mobile, une application de paiement dite « maître » est informée de cet événement par le contrôleur NFC. Elle gère alors la transaction de paiement en faisant appel à l'application concernée. Cette dernière envoie alors les commandes – correspondant à l'application de paiement sélectionnée – à la carte au travers du contrôleur NFC. L'application maître comprend un module chargé de l'interface utilisateur. Cette application est exécutée depuis le monde normal de la TEE sauf pour deux opérations : la vérification du porteur par saisie du code PIN ou par capture de la signature, et la confirmation du montant de la transaction. En effet, deux des attaques les plus redoutées dans le contexte du paiement mobile sont l'interception du code PIN (exemple : à l'aide d'une application espionnant les touches frappées au clavier avec un *keylogger*) et

la falsification du montant affiché de la transaction (exemple : afficher un montant de 10€ alors que la transaction est menée pour 100€). Pour ces raisons, ces deux opérations sont basées sur des routines sécurisées dans la TEE. Elles participent à la mise en œuvre de ce que l'on appelle « *Secure Mobile PIN* » et « *What You See Is What You Pay* » (WYSIWYP), traduits littéralement par : « code PIN sur mobile sécurisé » et « ce que tu vois est ce que tu payes ». Pour cela, les périphériques tels que le clavier et l'écran sont connectés au micro-processeur de l'équipement et leurs accès sont sécurisés et n'autorisés qu'aux applications de la TEE.

Pour s'assurer de l'intégrité de l'application maître, on calcule une empreinte de son code que l'on place dans le Secure Element. Chaque fois qu'elle est lancée, une routine sécurisée de la TEE vérifie l'intégrité de l'application maître en comparant l'empreinte actuelle de son code et l'empreinte de référence.

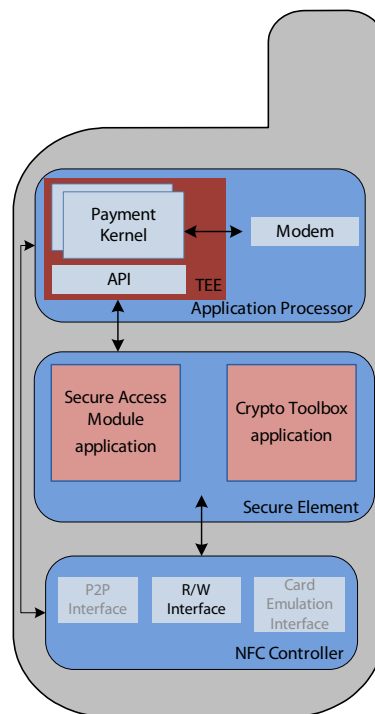


FIGURE 3.9 – Les applications de paiement dans le processeur d'applications et la cryptographie dans le Secure Element

3.3.5.3 Tous les composants dans le Secure Element

Cette approche est similaire à la première, sauf que les composants ne sont pas embarqués dans un module externe mais dans le Secure Element. Cette solution est illustrée à la figure 3.10. Les applications de paiement ainsi que le SAM sont hébergées

de manière sécurisée et exécutées depuis le Secure Element. Cette architecture permet de tirer parti des architectures de Secure Element existantes telles que celles s'appuyant sur GlobalPlatform. C'est un point essentiel pour le chargement et la personnalisation des applications de paiement. Ces applications et le SAM sont hébergées dans un domaine de sécurité séparé ayant son propre jeu de clés et son propre mode de gestion.

Lorsqu'une carte est détectée dans le champ radio-fréquence du mobile, le Secure Element gère directement la transaction de paiement. L'application de paiement concernée est lancée et la transaction se déroule. A certains moments clés (vérification du porteur, demande d'autorisation à l'émetteur, fin de la transaction...), le Secure Element envoie des événements à l'application située dans le processeur d'application afin de modifier les informations affichées et, le cas échéant, interagir avec le porteur.

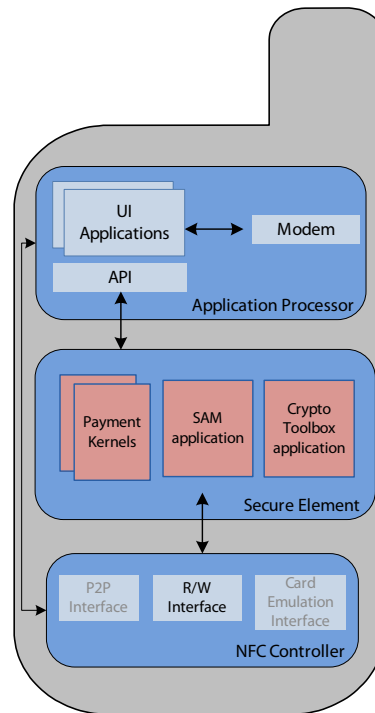


FIGURE 3.10 – Les applications de paiement et la cryptographie dans le Secure Element

3.3.6 Analyse des solutions proposées

Dans cette section, nous discutons les trois architectures proposées précédemment, les comparons selon des critères objectifs et synthétisons leurs avantages et limitations.

Les critères choisis pour comparer sont le niveau de sécurité offert par la solution, la certificabilité et l'effort d'intégration de la solution pour les manufacturiers.

3.3.6.1 Tous les éléments dans un module externe

Sécurité : cette solution offre un fort niveau de sécurité car les composants sont situés en-dehors de l'équipement mobile, dans un module certifié. Une telle solution obtiendrait aisément une certification sécuritaire telle que les Critères Communs [76] car la cible d'évaluation (*Target of Evaluation*, TOE) est simple.

Intégration : cette solution souffre de plusieurs difficultés d'intégration. En effet, jusqu'au jour où les équipements mobiles auront un port pour les accessoires standardisés, le module doit être implémenté pour tous les modèles. De plus, de nouvelles APIs doivent être ajoutées à la couche logicielle.

Analyse : cette solution permet de transformer un équipement mobile en un terminal de paiement sans contact NFC juste en branchant un module externe. Elle requiert un effort d'intégration non négligeable car elle utilise un connecteur propriétaire et nécessite de nouvelles APIs.

3.3.6.2 Processeur d'applications et Secure Element

Sécurité : cette solution est basée sur des concepts et composants hautement sécurisés mais l'intégration d'une TEE dans un mobile n'est pas encore très mature du côté des manufacturiers d'équipements mobiles. La certification semble également être un point difficile car la cible d'évaluation est le système d'exploitation entier. Des initiatives telles que la certification par composition [77] sont actuellement à l'étude dans les organismes de standardisation.

Intégration : la TEE est un concept de sécurité prometteur mais souffre encore d'un manque d'intégration par les manufacturiers.

Analyse : en combinant un système d'exploitation de confiance et un composant électronique sécurisé, cette solution offre d'un niveau de sécurité élevé. Ce type d'architecture peut aider à l'adoption du paiement mobile, notamment en sécurisant l'entrée du code PIN sur le clavier de l'équipement mobile. Néanmoins, comme la technologie TEE n'est pas encore mature, cette solution n'est pas viable à court terme. Mais, il semble évident qu'elle sera l'une des solutions les plus viables à long terme.

3.3.6.3 Tous les composants dans le Secure Element

Sécurité : dans cette solution, tous les composants sont hébergés par le Secure Element, une combinaison de matériel et logiciel hautement sécurisé de par sa conception. Ainsi, les applications de paiement et le SAM sont localisées dans l'endroit le plus sécurisé de l'équipement mobile et héritent de ses propriétés de sécurité.

Intégration : cette approche est très sécurisée mais implique un gros effort d'intégration notamment pour le lien entre le Secure Element et le contrôleur NFC. En effet, l'un des modes de fonctionnement du NFC est l'émulation de carte en s'appuyant, entre autres, sur le Secure Element. Dans le mode de fonctionnement présenté ici, le Secure Element doit émuler un terminal, *i.e.* être l'initiateur de la transaction et envoyer des commandes à la carte au lieu de, comme une carte, accepter les commandes entrantes, faire des traitements en interne, et répondre. Ceci requiert de changer le fonctionnement du contrôleur NFC et du Secure Element. Une autre préoccupation concerne le temps de transaction. En effet, les Secure Element sont optimisés pour effectuer des opérations cryptographiques telles que la signature où le calcul d'empreinte mais ne le sont pas pour effectuer l'opération inverse, *i.e.* la vérification de signature et d'empreinte.

Analyse : avec un effort d'intégration conséquent sur le Secure Element en « mode terminal », cette solution peut être d'une grande valeur ajoutée pour la sécurisation de l'entrée du PIN et de l'affichage du montant. En effet, on peut imaginer que le Secure Element sera totalement intégré à la conception d'une nouvelle génération d'équipements mobiles sécurisés et aidera ainsi à l'adoption de la technologie NFC.

3.3.6.4 Étude comparative

Le tableau 3.2 compare les trois solutions et architectures exposées selon les critères suivants : la facilité d'intégration dans un équipement mobile, le niveau de sécurité et la certifiabilité. Une note globale synthétise la solution. Les notes vont de ★ à ★★★★★ (le symbole ★ représente une demi-étoile) et sont assignées, de façon subjective, selon l'étude comparative réalisée ci-dessus.

3.3.7 Discussion

Nous avons présenté dans cette section trois architectures pour implémenter un terminal de paiement électronique mobile sans contact avec un équipement mobile

	Intégration	Sécurité	Certification	Global
Module externe	★★	★★★★★	★★★☆	★★★☆
PA & SE	★★	★★★★	★★★★☆	★★★★
Secure Element	★	★★★★★	★★★☆	★★★☆

TABLE 3.2: Comparaison des solutions proposées

utilisateur doté de la technologie NFC. Les trois solutions proposées présentent un bon niveau de sécurité et nécessitent un effort d'intégration plus ou moins grand par les fabricants d'équipements mobiles. Comme elle permet de saisir un code PIN et d'afficher les informations de la transaction de manière sécurisée, nous pensons que la solution combinant les applications de paiement dans le processeur d'applications (s'appuyant sur les services de la TEE) et le Secure Element est la solution la plus viable à long terme. Une telle architecture devrait aider à l'adoption de la technologie NFC en général et du paiement mobile en particulier.

Regardons maintenant cette solution du point de vue de la certification PCI DSS en général, et plus particulièrement PA-DSS et PCI-PTS. Dans le cas qui nous intéresse, l'acquisition de transaction de paiement, PCI recommande deux choses :

l'acquisition sécurisée des données : par crainte d'être interceptées par une application malveillante, les données doivent transiter par un composant sécurisé aussitôt acquises ;

le chiffrement des données de bout en bout : il est recommandé pour assurer une protection maximale des données (mais aussi pour alléger la responsabilité du commerçant au regard de PCI) de procéder à un chiffrement des données sensibles de bout en bout entre le mobile et la passerelle d'acquisition.

Ces exigences sécuritaires sont remplies par la solution consistant en un module externe. La solution avec tous les composants dans le SE et celle combinant les applications de paiement dans le processeur d'applications et le SE peuvent répondre à l'exigence de chiffrement de bout en bout par le biais d'un chiffrement dans le SE avec une clé partagée mais ne peuvent répondre à l'exigence d'acquisition sécurisée des données.

Afin de permettre à ces deux dernières solutions de répondre aux recommandations sécuritaires PCI DSS, nous avons spécifié et développé au sein d'INSIDE Secure, une technologie en cours de dépôt de demande de brevet permettant de sécuriser les données dites sensibles dès leur acquisition sur l'interface sans contact et d'en assurer

le chiffrement de bout en bout. Pour des raisons de confidentialité, cette solution ne peut pas être divulguée dans ce mémoire.

3.4 Aide au déploiement de services mobiles sans contact

3.4.1 Introduction

Empiriquement, le déploiement des applications sur un SE s'appuie sur la connaissance *a priori* de la « cible » par le TSM. Même si cette méthode s'avère efficace et est éprouvée, elle présente des limites notamment dans le cas de déploiements massifs. En effet, on peut présumer que dans les années à venir, les TSMs auront la charge de gérer des millions de SE et devront alors déployer de puissants et fiables systèmes de gestion de SE comparables à ce qui est mis en œuvre dans les domaines bancaires et télécommunications (CMS, *Card Management Systems*). Les CMS sont adaptés à l'industrie de la carte plastique car ce facteur de forme est statique, les applications contenues sont pré-chargées et personnalisées par le bureau de personnalisation avant d'être émises, *i.e* envoyées à leurs porteurs. La situation est différente avec un SE car les applications peuvent y être chargées et personnalisées après l'émission. Ces mécanismes sont appelés « post-installation » et « post-personnalisation ». Ils impliquent que les CMSs soient capables d'être interrogés, d'analyser la configuration du SE et de générer les scripts de déploiement en temps réel! Pour pallier cette contrainte forte, une solution consiste à modéliser le SE. En effet, si on sépare le SE en couches on peut distinguer :

- un système d'exploitation (Java Card),
- une plateforme responsable de la gestion du contenu applicatif et du cycle de vie (GlobalPlatform),
- et des applications.

La modélisation d'un SE consisterait donc aux éléments suivants :

- un modèle de données d'une plateforme GlobalPlatform conforme aux spécifications [51],
- un méta-modèle définissant un SE en décrivant :
 - les fonctionnalités GlobalPlatform supportées par le SE,
 - les fonctionnalités propriétaires liées à l'implémentation de GlobalPlatform par le fabricant du système d'exploitation,
 - le contenu applicatif du SE (byte code des paquets Java Card, applications, ...).

La modélisation GlobalPlatform est présente à la fois du côté de l'émetteur et du côté du TSM. Une fois que l'émetteur a instancié les données dans le modèle et obtenu un méta-modèle, il distribue ce dernier au TSM. Ce dernier peut alors « injecter » le méta-modèle dans le modèle GlobalPlatform et obtenir ainsi une modélisation du même SE. Puis, grâce une propriété de réflexivité, le modèle du SE peut générer automatiquement les scripts de déploiement d'une application donnée sur ce SE. Une autre solution consiste à localiser le méta-modèle dans le SE et à inviter le TSM à le récupérer directement.

Dans les sections suivantes, nous présentons les différents éléments mis en œuvre afin de réaliser ce dernier scénario illustré par la figure 3.11. Nous présentons les outils utilisés pour créer la modélisation d'un SE GlobalPlatform (zones A et B) et le framework qui permet sa manipulation (zone B). Nous montrons que modéliser un SE en utilisant une ontologie nous permet de représenter les concepts statiques de celui-ci ainsi que son comportement dynamique et temps-réel. Enfin, nous étudions son application pour l'aide au déploiement de services mobiles sans contact, en particulier, la localisation des données de configuration sur le SE (zone C) et les actions requises par le TSM (zone D).

3.4.2 État de l'art des méthodes de modélisation

3.4.2.1 Propriétés attendues

Avant d'étudier les différentes méthodes de modélisation, listons les propriétés attendues d'une méthode qui répondrait pleinement à nos besoins :

dynamique, le modèle doit permettre d'instancier dynamiquement des classes.

paramétrable, le modèle doit permettre l'injection de règles de manière dynamique.

sémantique, la méthode de modélisation doit offrir la possibilité de donner un sens fort aux relations entre les objets et aux objets eux-mêmes.

accessible par un langage de programmation, l'outil de modélisation doit offrir une bibliothèque permettant d'être intégrée à un programme utilisateur et de manipuler le modèle via ce programme.

interrogeable, l'outil de modélisation doit offrir un langage d'interrogation permettant de sélectionner des instances ou de renseigner sur la présence ou non d'une certaine instance répondant à des critères précis.

3.4.2.2 Introduction

Le terme « modèle de données » peut avoir deux significations [78] :

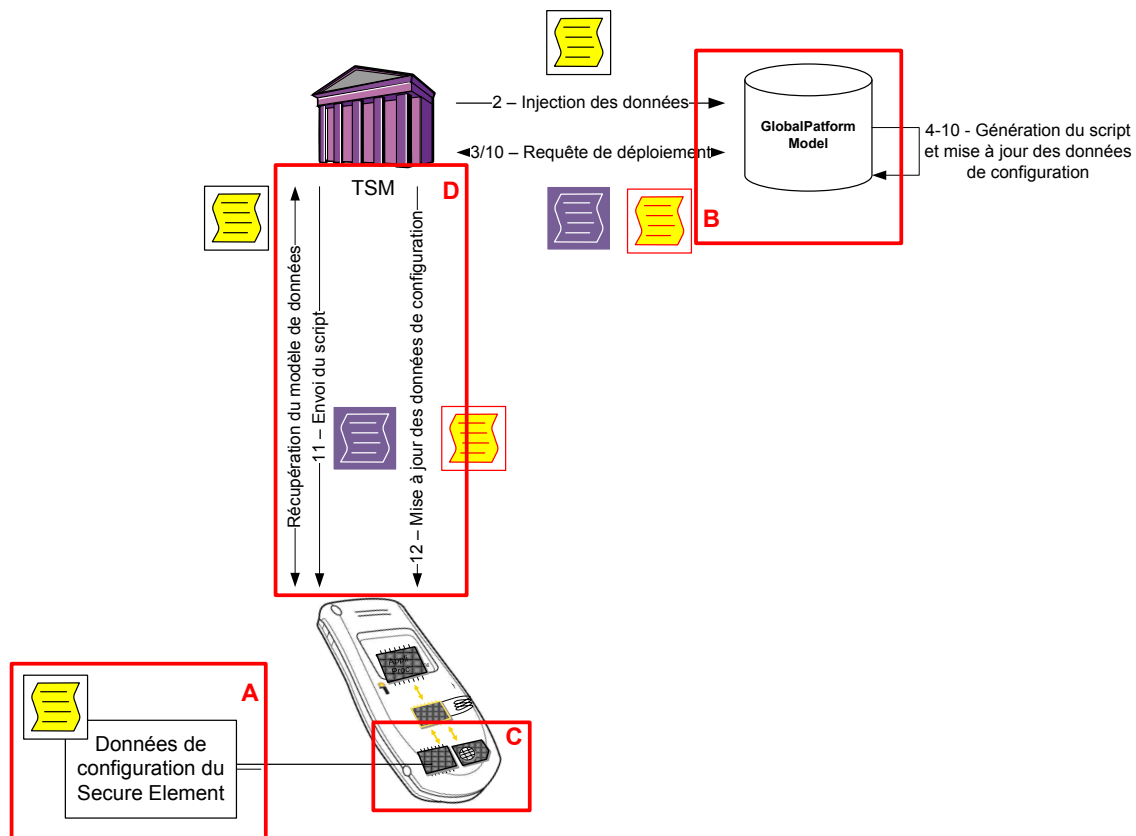


FIGURE 3.11 – Éléments mis en œuvre dans la solution proposée

- un modèle de données théorique, *i.e.* une description formelle ou un modèle mathématique.
- un modèle de données *instance*, *i.e.* qui applique un modèle de données théorique pour créer un modèle de données *physique*.

Un modèle de données théorique a trois composants principaux :

- une partie structurelle qui définit la structure des données utilisées pour représenter les objets,
- une partie intégrité qui fournit un ensemble de règles et contraintes sur les données afin de garantir l'intégrité de la structure du modèle,
- une partie manipulation qui fournit un ensemble d'opérations permettant de créer, sélectionner, mettre à jour et supprimer les données.

Selon l'ANSI, un modèle de données *instance* est de l'un des trois types suivants [79] :

- modèle conceptuel, qui définit la sémantique d'un domaine auquel s'applique le modèle. Cela consiste en des entités représentant des éléments du domaine et des

assertions sur les relations entre une paire d'entités provenant du modèle. Un modèle conceptuel spécifie les faits et propositions qui peuvent être exprimées à partir du modèle. En ce sens, il définit les expressions autorisées par le biais d'un langage dont la portée est limitée par la portée du modèle.

- modèle logique, qui définit la sémantique mais telle que représentée par une technologie de manipulation de données spécifiées. Par exemple, un modèle logique consisterait en des tables, colonnes, classes orientées objet ou encore des tags XML.
- modèle physique, qui décrit la façon dont les données sont stockées.

Dans cet état de l'art, nous nous intéressons aux méthodes permettant de définir un modèle de données conceptuel. Comme l'a souligné Bubenko [80], des centaines des méthodologies plus ou moins proches ont été publiées et on estime que plusieurs milliers d'approches différentes sont actuellement utilisées. Devant la pléthore de méthodologies, nous restreignons notre étude aux trois grandes familles que sont les modèles relationnels, orientés objet et sémantique. Nous étudions également les précédentes tentatives de modélisation d'un SE GlobalPlatform présentes dans la littérature ou dans l'industrie puis concluons en présentant la méthode utilisée au vu de cet état de l'art.

3.4.2.3 Le modèle relationnel

Le modèle relationnel fut introduit par E.F Codd en 1970 [81]. Chercheur chez IBM à la fin des années 1960, il étudiait de nouvelles méthodes pour gérer de grandes quantités de données car les outils de l'époque n'étaient pas satisfaisants. Mathématicien de formation, il utilisa des propriétés de la théorie des ensembles pour résoudre des difficultés telles que la redondance des données, l'intégrité des données ou l'indépendance de la structure de la base de données avec sa mise en œuvre physique [82]. Les travaux de Codd aboutirent au développement d'un premier prototype de système de gestion de base de données relationnelles (SGBDR). Depuis, cette technologie a évolué et a été largement adoptée par l'industrie et en 1987, le langage SQL (Structured Query Language) a été standardisé [83].

Le modèle relationnel est basé sur une organisation des données en tables. Voici quelques définitions des éléments de base d'un modèle relationnel [84, 85]. Les colonnes ou « attributs » représentent des attributs tels que le nom ou l'âge d'une personne. Les lignes ou « tuples » représentent des instances uniques telles que des personnes ayant un nom et un âge donnés. On appelle domaine d'un attribut l'ensemble fini ou infini, de ses valeurs possibles. Une relation est un sous-ensemble du produit

cartésien de n domaines ($n > 0$). Elle est représentée sous la forme d'un tableau à deux dimensions dans lequel les n attributs correspondent aux titres des n colonnes. La clé primaire d'une relation est l'attribut, ou l'ensemble d'attributs, permettant de désigner de manière unique un tuple. Une clé étrangère est une clé faisant référence à une clé appartenant à une autre table. La manipulation des tuples d'une table se fait à l'aide d'opérations sur les ensembles, en particulier celle de produit cartésien. Le produit cartésien d'un ensemble de domaines D_i , noté $D_1 * D_2 * D_3 * \dots * D_n$ est l'ensemble des n -uplets (tuples) $\langle V_1, V_2, \dots, V_n \rangle$ tels que V_i appartient à D_i . Il existe deux types d'opération de base :

- les opérations unaires, qui consistent à éliminer des lignes ou des colonnes d'une table ;
- les opérations ensemblistes, qui consistent à effectuer un recoupement entre plusieurs tables.

Les opérations unaires sont au nombre de deux :

- la *projection*, qui consiste à créer une table à partir d'une autre en ne gardant que les colonnes spécifiées ;
- la *restriction* ou sélection, qui consiste à créer une table à partir d'une autre en ne gardant que les lignes lesquelles une ou plusieurs colonnes correspondent aux critères donnés.

Parmi les opérations ensemblistes on retrouve :

- l'*union*, qui est une table contenant l'ensemble des tuples des deux tables unies. Cependant, les deux tables doivent respecter le même schéma, *i.e.* elles doivent avoir les mêmes attributs ;
- la *différence*, qui est une table contenant l'ensemble des tuples appartenant à une table mais pas à une seconde. Les deux tables doivent respecter le même schéma ;
- le *produit cartésien*, qui est une table dans laquelle on retrouve la concaténation de l'ensemble des tuples d'une ligne d'une table à ceux de l'autre table, et ce pour chaque ligne.

Il existe une troisième catégorie d'opérations construites à partir des opérations de base, que nous appelons « les opérations dérivées ». On y retrouve :

- l'*intersection*, qui contient l'ensemble des tuples appartenant aux deux tables. Les deux tables opérantes doivent avoir le même schéma ;
- le *quotient*, qui contient l'ensemble des tuples qui concaténés à chaque tuple de l'une des tables fournissent des tuples appartenant à l'autre ;

- la θ -jointure (théta-jointure) selon une qualification Q , qui contient l'ensemble des tuples provenant du produit cartésien de deux relations et satisfaisant la qualification, *i.e.* la condition exprimée à l'aide des comparateurs \geq , \leq , $>$, $<$, $=$, \neq , \neg . On définit deux types de jointure particulières :
 - l'équi-jointure dont la qualification est une égalité entre deux colonnes ;
 - la jointure naturelle, équi-jointure sur des attributs de même nom associée à une projection.

Enfin, il existe des opérateurs de calcul tels que la somme ou le comptage. Ainsi, par l'utilisation de la théorie des ensembles, Codd a rendu possible la modélisation d'une banque de données dans des tables à deux dimensions et a permis une manipulation avancée de ces données.

3.4.2.4 Le modèle orienté objet

Avec la massive implémentation des bases de données relationnelles et le succès grandissant de la programmation orientée objet, il devint nécessaire de pouvoir faire persister les objets créés par un programme. Les techniques de la programmation orientée objet et des bases de données relationnelles durent alors cohabiter.

La partie logicielle d'un système, quel qu'il soit, étant composée soit de données soit de programmes, les méthodes de conception traditionnelles se focalisent soit sur les données soit sur les fonctionnalités du logiciel. Dans [86], les deux méthodologies de conception suivantes sont données comme les deux grandes « écoles de pensée » :

- la *conception orientée données*. Aussi connue sous le nom d'*ingénierie de l'information*, elle s'intéresse à l'analyse des besoins du système en données. Ces besoins guident ensuite la conception logicielle.
- la *conception orientée fonctionnalités*. Aussi appelée *analyse structurée*, elle est basée sur la décomposition des processus afin de réduire les fonctionnalités du système à un modèle hiérarchique [87]. Les besoins fonctionnels guident alors la conception logicielle.

Avec l'amélioration des méthodes et outils de développement logiciel, ces deux méthodes de conception ont évolué et abouti à la conception orientée objet ou modélisation orientée objet (MOO). MOO combine des éléments des méthodes de conception orientée données et orientée fonctionnalité. Elle décompose un système en objets comprenant à la fois des données sous formes d'attributs et des fonctionnalités sous formes de méthodes. Elle apporte ainsi une grande flexibilité de par l'interchangeabilité et la réutilisabilité des briques que constituent les objets. Il existera autant de configurations que de combinaisons possibles entre les objets.

3.4.2.4.1 Le diagramme de classes

Le diagramme de classes est l'équivalent pour le modèle orienté objet du modèle de données relationnelles. Il présente une vue statique du système et fait apparaître les trois grands concepts suivants [88] :

- les attributs, données liées à l'objet, avec leur visibilité (publique, protégé, privé) et multiplicité ;
- les méthodes, fonctionnalités réalisées par l'objet avec leur visibilité et direction des paramètres (entrant, sortant ou entrant et sortant) ;
- les relations entre les classes.

3.4.2.4.2 UML

Unified Modeling Language (UML) est né du besoin d'avoir une unification des méthodes de modélisation sur plusieurs domaines d'application. UML est basé sur les méthodes de BOOCH (de Grady Booch dans [89]), OMT (Object Modeling Technique, de Jim Rumbaugh dans [90]) et OOSE (Object-Oriented Software Engineering, de Ivar Jacobson dans [91]). Les objectifs de ce langage de modélisation sont les suivants [92] :

- représenter des systèmes en entier
- établir un couplage explicite entre les concepts et les artefacts exécutables
- prendre en compte les facteurs d'échelle
- créer un langage de modélisation utilisable à la fois par les humains et les machines

Les composants de base du langage UML sont :

- des éléments
 - structuraux (classes, interfaces, collaborations, cas d'utilisation ...)
 - d'état (machines d'état)
 - de groupe (paquetages, notes)
 - d'annotation (notes)
- des relations (dépendance, association, généralisation, réalisation)
- des diagrammes (de classes, d'objets, de cas d'utilisation, de séquence, de collaboration, d'état, d'activité, de composants, de déploiement)

Par le biais de ces composants de base et le méta-modèle qui définit lui-même le langage, UML est sémantiquement riche. Il permet pour chaque objet d'attribuer

un nom, une portée, une visibilité, un rôle dans sa relation avec un autre objet et une cardinalité. A partir de la version 2.0, UML intègre le langage OCL (Object Constraint Language), permettant d'exprimer des contraintes sur un modèle. La capacité d'OCL à parcourir les modèles permet de l'utiliser comme un langage de requête [93]. OCL peut aussi être utilisé pour raisonner sur un modèle UML afin de déterminer s'il satisfait certaines propriétés [94].

3.4.2.5 Le modèle sémantique

Le besoin d'un modèle sémantique est né dans les années 1970 d'un manque d'abstraction du modèle conceptuel par rapport au modèle physique. Jusqu'alors, le modèle conceptuel était extrêmement lié à l'implémentation physique des données et supportait très difficilement les concepts de relation, abstraction de données, héritage ... [95]. Ainsi, le modèle sémantique apporte une indépendance des données vis-à-vis de l'implémentation physique et permet d'exprimer des relations, dépendances ou associations [96]. Dans les sections suivantes, nous détaillons le premier modèle sémantique apparu (*i.e.* le modèle entité-relation) et un modèle sémantique très récent, les ontologies.

3.4.2.5.1 Le modèle entité-relation

Le modèle entité-relation (E-R) est apparu pour la première fois dans [97]. Il unifie les caractéristiques des modèles traditionnels (orientés objet) pour faciliter l'introduction d'informations sémantiques. Comme son nom l'indique, les composants de base sont l'entité et la relation. Les données sont alors vues comme une collection d'entités et de relations. E-R supporte la cardinalité et l'agrégation d'entités. Les entités peuvent être identifiées de manière unique par la valeur de ses attributs ou son type de relations avec d'autres entités [96].

3.4.2.5.2 Les ontologies

Dans [98], Parekh *et al.* donnent une définition très claire de ce que sont les ontologies : « Les ontologies sont conçues pour permettre une conceptualisation abstraite de l'information et un lexique de termes utilisés au sein de cette représentation. Elles donnent au domaine une sémantique et définissent les concepts du domaine et les relations existant entre eux. ». En d'autres termes, une ontologie permet de modéliser un ensemble de connaissances dans un domaine donné, réel ou imaginaire. Pour cela, les ontologies décrivent :

- des concepts ou classes ;
- des individus, instances des concepts ;
- des propriétés, caractéristiques des concepts ou d'autres propriétés ;
- des contraintes sur les concepts et leurs propriétés ;
- ...

Une multitude d'outils ont vu le jour pour manipuler les ontologies. Afin de développer un langage standardisé, le *World Wide Web Consortium* (W3C) a créé en Novembre 2001 un groupe de travail (WebOnt) rassemblant les acteurs du domaine. Le travail de ce groupe a résulté en la recommandation *OWL Web Ontology Language* en Février 2004. Une ontologie OWL consiste en un document XML. Elle peut être interrogée et offre une capacité d'inférence sur les individus créés afin de les classer selon les propriétés exprimées dans l'ontologie. Pour manipuler une ontologie OWL, il existe plusieurs outils dont *Jena Framework* qui offre un moteur d'inférence et un format d'inférence de règles intégrés.

Dans une ontologie OWL, les concepts sont représentés par des classes. L'objet résultant de l'instantiation d'une classe est appelé un individu. Par défaut, 2 classes ne sont pas disjointes, *i.e.* un individu peut être une instance de ces 2 classes et même plus. Pour permettre à un individu d'être une instance d'une seule classe OWL permet la déclaration de classes disjointes. Les relations entre les classes et entre les individus sont représentées à l'aide de propriétés qui peuvent être de deux types : propriété d'objet (*Object Property*) et propriétés de type de donnée (*DataType Property*). La propriété d'objet définit la relation existante entre deux concepts alors que la propriété de type de donnée décrit l'assignation d'une donnée à une certaine valeur ou plage de valeurs. Une propriété lie un individu d'un domaine (*domain*) à un individu cible d'un autre domaine (*range*). OWL permet également d'appliquer des restrictions de cardinalité et de quantification des propriétés d'objet. Les restrictions de quantification sont :

- *some* ou *existential* : un individu participe dans *au moins une* relation au travers d'une propriété la liant avec un individu d'une classe donnée ;
- *only* ou *universal* : un individu participe des relations au travers d'une propriété la liant *seulement* avec les individus d'une classe donnée.

Les cardinalités existantes sont les suivantes :

- *minimum* : pour une propriété donnée, elle spécifie le nombre *minimum* de relations auxquelles un individu peut participer ;
- *maximum* : pour une propriété donnée, elle spécifie le nombre *maximum* de relations auxquelles un individu peut participer ;
- *exactly* : pour une propriété donnée, elle spécifie le nombre *exact* de relations

auxquelles un individu peut participer ;

Ces restrictions assurent un grand contrôle sur l'affectation de propriétés à des classes et individus.

3.4.2.6 Modélisation d'un SE

On trouve dans la littérature plusieurs implémentations d'un modèle de carte à puce ou de SE. La plupart d'entre elles ont pour but de réaliser un audit sécuritaire ou de faire une vérification formelle d'une application. Par exemple, utilisant le modèle objet, Jürjens dans [99] utilise des diagrammes de classes et des diagrammes d'état UML pour modéliser de façon formelle et réaliser un audit sécuritaire d'une application de porte-monnaie électronique. Utilisant des méthodes formelles, Haneberg *et al.*, dans [100], présentent une méthode basée sur ASM (*Abstract State Machines*) et JML (*Java Modeling Language*) pour modéliser la sécurité de la communication entre une carte à puce et un lecteur. JML a aussi été utilisé dans le projet Européen *VerifiCard* pour réaliser des vérifications formelles d'applets commerciales [101]. De la même manière, Bouquet *et al.*, dans [102], montrent l'utilisation de la méthode *B* pour générer des cas de tests pour réaliser des tests fonctionnels de plateformes ou d'applets. Enfin, dans [103], Béguelin utilise également la méthode *B* afin d'effectuer une vérification formelle de la plateforme GlobalPlatform.

Dans l'industrie, on retrouve une implémentation majeure publiée par GlobalPlatform en 2003 dans [104]. Cette modélisation présentée sous forme XML a pour but de définir la structure et le contenu des données fournies au systèmes de gestion des cartes. Elle définit les concepts de base de la spécification GlobalPlatform tels qu'un canal sécurisé, une application, un privilège, un cycle de vie ... De plus, cette représentation est accompagnée d'un langage de scripting [105] permettant de désérialiser un profile en objets concaténés dans un script et d'exécuter ce dernier.

3.4.2.7 Discussion

Dans cette section, nous avons présenté les principales méthodes de modélisation existantes et les travaux de l'état de l'art sur la modélisation d'un SE. Le tableau 3.3 présente la façon dont les trois principales méthodes étudiées répondent aux propriétés attendues exprimées à la section 3.4.2.1 : dynamique, paramétrable, sémantique ; accessible par une API et interrogeable.

Le modèle relationnel permet la création dynamique d'objets, il existe des APIs permettant de créer une couche d'abstraction et il supporte les requêtes. En revanche,

Modèle	dynamique	paramétrable	sémantique	présence API	interrogeable
Relationnel	Oui	Non	Non	Oui	Oui
Orienté objet	Oui	Non	Oui	Oui	Oui
Sémantique	Oui	Oui	Oui	Oui	Oui

TABLE 3.3: Comparaison des modèles étudiés selon les propriétés attendues

il ne supporte pas l'injection dynamique de règles ou de contraintes fonctionnelles et souffre d'une pauvreté sémantique [106].

Le modèle orienté objet répond à toutes les propriétés sauf celle d'être paramétrable par l'injection dynamique de règles. Certes, si l'on prend l'exemple d'OCL pour UML, il est possible d'effectuer de la programmation « par contrat » et ainsi vérifier que des données répondent à des spécifications, mais ces règles doivent être écrites à l'avance et de manière statique.

Héritant de nombreuses années de recherche, de spécifications et de standardisation des modèles de données, les modélisations sémantiques telles que les ontologies sont les modèles les plus adaptés à notre besoin exprimé à la section 3.4.2.1. En particulier, OWL tire profit des travaux récents menés dans le domaine du Web Sémantique. Il existe notamment l'outil libre d'accès *Jena* qui est tout à fait adapté à nos besoins : il permet par programmation de créer des objets, de les manipuler, d'interroger le modèle et de l'inférer.

Pour toutes ces raisons, nous avons opté pour le langage d'ontologie OWL et l'outil *Jena* pour la modélisation d'un SE implémentant les spécifications GlobalPlatform. De plus, nous nous efforçons d'encapsuler la modélisation proposée par GlobalPlatform en 2003 afin de l'enrichir grâce aux technologies actuelles et d'apporter à nos travaux une application concrète dans l'industrie.

3.4.3 Présentation du framework

Afin de modéliser un SE GlobalPlatform et mettre cette modélisation en œuvre, nous avons développé un framework dans le langage C# et suivant une architecture en couches. Les couches de ce système sont : la couche des connaissances, la couche applicative et la couche de décision. Le framework est présenté à la figure 3.12. Dans les sections suivantes, nous décrivons les composants des différentes couches et leurs interactions.

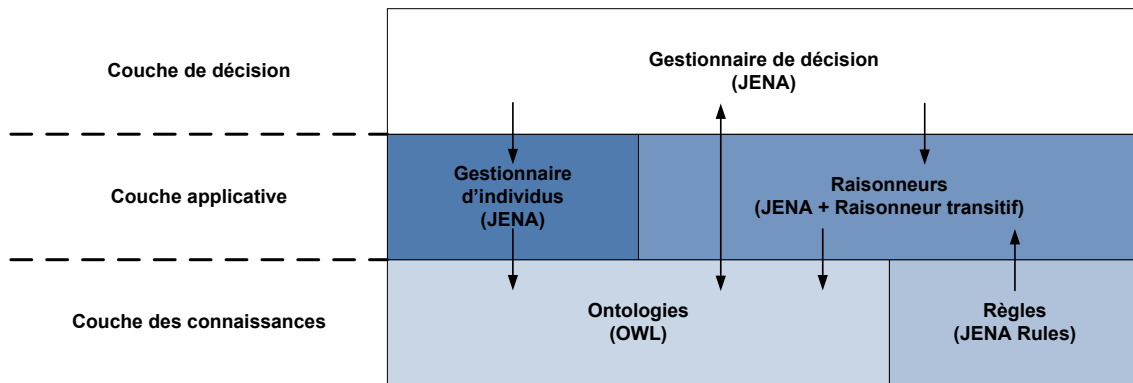


FIGURE 3.12 – Framework de modélisation d'un SE

3.4.3.1 L'ontologie

L'ontologie est le composant principal du framework. Elle est utilisée pour représenter les concepts du SE. Notre ontologie est écrite dans le langage OWL (Web Ontology Language) construit sur le modèle de données RDF (Resource Description Framework) [107]. OWL est basé sur une sémantique formelle et déclarative définie par une syntaxe rigoureuse. Des programmes externes, appelés « raisonneurs », peuvent ainsi à la fois vérifier la cohérence de l'ontologie et découvrir dynamiquement et implicitement des connaissances de cette ontologie. Une telle particularité rend ce langage très utile pour les outils d'aide à la décision.

La classe centrale de l'ontologie GlobalPlatform est la classe *Policy* (politique), concept inspiré des travaux conduits sur la gestion de politiques à base d'ontologies dans des projets tels que SOUPA [108], Rei [109] ou KAOS [110, 111]. On assigne un individu à une politique – sur lequel elle a le contrôle – par le biais de la propriété d'objet *Controls*. Cette propriété, en prenant la valeur *Permits* ou *Forbids*, permet de déterminer si une action est autorisée ou refusée. Les deux propriétés ont un *range* bien distincts : *PermittedAction* et *ForbiddenAction*. Ceci implique que la politique autorise ou interdit une action. Cependant, comme les concepts *PermittedAction* et *ForbiddenAction* ne sont pas disjoints, une autorisation peut être à la fois autorisée et interdite ce qui donne lieu à un arbitrage allant la plupart du temps vers l'interdiction de l'action. Nous utilisons abondamment le mécanisme de *Policy* pour modéliser le comportement dynamique du SE. Par exemple, comme expliqué à la section 3.4.5.1, nous déterminons à l'aide de règles si l'opération d'installer une application donnée est autorisée ou pas.

3.4.3.2 Les règles

Comme nous l'avons vu à la section précédente, un concept *Policy* peut autoriser ou interdire un concept *Action*. Une telle décision est rendue possible par l'application de règles à l'ontologie. Les règles utilisées dans notre framework sont écrites dans le langage *JenaRules* intégré au framework Jena. Pour construire une règle, JenaRules utilise deux types d'atomes [112] :

- des triplets : structures de type (sujet, prédicat, objet), où le sujet, le prédicat et l'objet sont des nœuds (exemple : `(?x rdf:type ont:Action)`);
- des primitives du langage JenaRules : prédicats de type booléen tels que *notEqual* ou *noValue*.

Un exemple de règle extrait de [113] est donné ci-dessous. Dans cet exemple, il est déclaré que tout conducteur ayant un « Driver School Certificate » (traduit par certificat d'aptitude à la conduite) et n'ayant pas eu d'accident est éligible à une assurance.

```
@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
@prefix ex: http://example.com/
@prefix xs: http://www.w3.org/2001/XMLSchema#
[eligibleDriver: (?d rdf:type ex:EligibleDriver)
<-
(?d rdf:type ex:Driver)
(?d ex:certificateType ex:DriverSchoolCertificate)
(?d ex:accidentsNumber "0"^^xs:integer)]
```

3.4.3.3 Les raisonneurs sémantiques

La particularité clé des ontologies est leur capacité à être « manipulées » par des raisonneurs sémantiques. Un raisonneur sémantique est un moteur d'inférence capable de classifier de manière logique des individus à partir d'axiomes. Par exemple, à partir d'une ontologie décrivant les différentes espèces d'oiseaux (couleur, taille, forme du bec ...), un raisonneur sémantique peut classifier un individu que l'on lui soumettrait afin de déterminer son espèce. Les raisonneurs transitifs sont utilisés pour réaliser la fermeture transitive des classes (et des propriétés car OWL permet l'héritage des propriétés). Les raisonneurs logiques tels que *Pellet* [114] sont également capables de vérifier la cohérence de l'ontologie, *i.e.* vérifier que chaque classe peut être instanciée. Ils procèdent également à la classification des concepts (classes et individus). Dans notre framework, deux raisonneurs sont appelés successivement. Un raisonneur transitif est tout d'abord lancé afin de valider l'ontologie et construire un modèle classifié. Ensuite, un raisonneur de règles générique intégré à Jena est lancé

afin d'inférer de nouveaux éléments de connaissance, en particulier prendre la décision via la *Policy* d'autoriser ou d'interdire une *Action*. Le raisonneur générique de Jena supporte trois modèles d'exécution, respectivement *forward*, *backward* et *hybrid*. Dans notre cas, nous utilisons le modèle *forward* basé sur l'algorithme standard RETE [115]. Le modèle *backward* est un modèle de programmation logique. Il peut résoudre des problèmes récursifs telle que la fermeture transitive pour éviter des boucles infinies. Le modèle *hybrid* consiste à utiliser une ou plusieurs règles de type *backward* avant une règle de type *forward*.

3.4.3.4 La gestion des individus de l'ontologie

Les règles contraignent le raisonneur afin qu'il classe des actions en autorisées ou interdites. Pour pouvoir faire cela, les individus correspondant aux actions doivent être présents dans l'ontologie. Dans notre framework, le gestionnaire d'individus est responsable de la création d'au moins trois individus et deux propriétés d'objets spécifiques. Le premier est l'action contrôlée par la politique. Les deux autres individus sont l'acteur ou entité extérieure qui réalise l'action et la cible de l'action. Le gestionnaire d'individus est également responsable de la création des deux triplets RDF suivants : `<action hasActor actor>` et `<action hasTarget object>`.

3.4.3.5 La prise de décision

Nous retrouvons au sommet de l'architecture de notre framework, le gestionnaire de décision. Tout d'abord, le gestionnaire de décision est en charge de l'initialisation de l'architecture. Il charge l'ontologie et les règles grâce aux méthodes `loadModel()` et `loadRules()`, et crée les raisonneurs par le biais de la méthode `createReasoners()`. Ensuite, il exécute les raisonneurs sur l'ontologie (`launchReasoners()`). Lorsque qu'une opération de gestion de contenu est effectuée sur le modèle, une requête est créée par le framework et le gestionnaire de décision interroge l'ontologie via la méthode `makeRequest()`. Le moteur de requête est basé sur SPARQL (Simple Protocol and RDF Query Language) [116] qui autorise quatre types de requête : `SELECT`, `ASK`, `CONSTRUCT` et `DESCRIBE`. Dans notre framework, nous utilisons les deux premières. `SELECT` permet d'extraire des données de l'ontologie tandis que `ASK` retourne un résultat booléen. Enfin, la méthode `makeDecision()` est invoquée afin que le gestionnaire de décision retourne la décision inférée par le raisonneur.

3.4.4 Présentation des outils

Dans cette section, nous présentons quelques outils utiles pour notre problème.

3.4.4.1 Protégé

Pour créer l'ontologie GlobalPlatform, nous avons utilisé l'outil Protégé de l'Université de Stanford [117]. Protégé est un éditeur à code source libre écrit en Java qui permet la lecture, l'édition et la sauvegarde d'ontologies dans la plupart des formats d'ontologie : RDF, RDFS (RDF Schema [118]), OWL, etc. Nous l'avons préféré à d'autres éditeurs tels que Hozo [119], OntoStudio [120] et Swoop [121] pour sa gratuité et sa convivialité grâce à son interface utilisateur simplifiée illustrée à la figure 3.13.

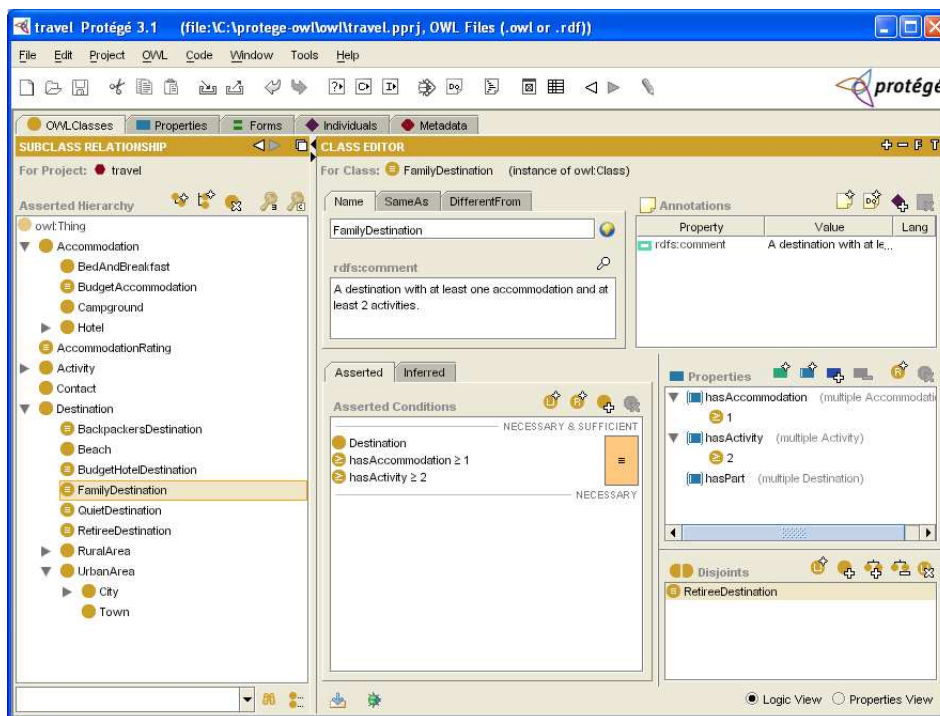


FIGURE 3.13 – L'interface utilisateur de l'outil Protégé

3.4.4.2 Le framework Jena

Jena [122] est un framework fournissant une collection d'outils et de bibliothèques Java pour le développement d'application pour le Web Sémantique. Il inclut un environnement pour la manipulation des formats RDF, RDFS, OWL, SPARQL [123] ainsi qu'un moteur d'inférence basé sur une logique de règles. Jena étant écrit en Java et nos développements réalisés en .NET, nous avons dû réaliser un portage de Jena vers Java. Pour cela, nous avons utilisé l'outil IKVM .NET [124]. Dans les sections suivantes, nous désignons par « framework GlobalPlatform » le code développé (basé sur Jena) pour créer et manipuler l'ontologie d'un SE GlobalPlatform.

3.4.5 Modélisation en couches du SE

3.4.5.1 La couche des connaissances : modélisation de GlobalPlatform

La modélisation de la spécification GlobalPlatform est réalisée grâce à une ontologie OWL, cœur de notre framework. Elle représente avec des classes et sous-classes RDFS les concepts décrits par la spécification :

- gestion des cycles de vies de la carte, des applications et du code source placé en mémoire ;
- les privilèges d’une application ou d’un domaine de sécurité ;
- le contenu applicatif : le code source, les applications et leurs AIDs respectifs
- les opérations de gestion de contenu : suppression, extradition, installation, chargement et mise à jour du registre ;
- les composants élémentaires du *runtime* GlobalPlatform : le registre GlobalPlatform et le *Trusted Framework*.

L’ontologie GlobalPlatform, centrée sur la classe `CCMOperation`, est représentée à la figure 3.14. Un individu du type `CCMOperation` représente une opération de gestion de contenu applicatif (*Card Content Management*, CCM), *i.e* un chargement, une suppression, une installation ou une mise à jour de l’entrée dans le registre d’un paquetage ou d’une application. Pour des raisons de lisibilité, seules les propriétés d’objet, qui définissent les relations entre deux concepts, sont représentées.

Afin de modéliser le comportement dynamique d’un SE GlobalPlatform, telle que l’installation d’une nouvelle application, l’ontologie est régie par un ensemble de règles. Ces règles sont injectées dans le raisonneur et conditionneront le résultat de la classification. Dans notre framework, nous avons défini un ensemble de trois fichiers de règles décrivant les domaines suivants :

- le domaine *des spécifications* : ces règles décrivent le comportement dynamique d’une implémentation GlobalPlatform – tel que défini par la spécification – lors des opérations de gestion de contenu applicatif. Par exemple, ces règles s’assurent qu’une opération de gestion de contenu (représentée par un individu de type `CCMOperation`) ne peut être permise que s’il existe un canal sécurisé (représenté par un individu de type `SecureChannel`) liant un entité extérieure au SE (représenté par un individu de type `OffCardEntity`) et le domaine de sécurité cible (représenté par un individu de type `SecurityDomain`) et qu’ils partagent la même clé secrète (valeur assignée à un individu de type `Key`).
- le domaine de l’implémentation du SE : ces règles décrivent le comportement du SE dû à l’implémentation de la plateforme par le fabricant. Par exemple, ces règles définissent la manière dont une fonctionnalité de la spécification GlobalPlatform a été implémentée par le fabricant.

- le domaine du domaine d'application : cet ensemble de règles définit des contraintes liées au domaine d'application. Par exemple, dans le domaine du paiement, le paquetage d'un domaine de sécurité doit avoir un AID spécifique et la création d'un domaine de sécurité supplémentaire à partir de ce même paquetage doit être effectuée d'une manière spécifique.

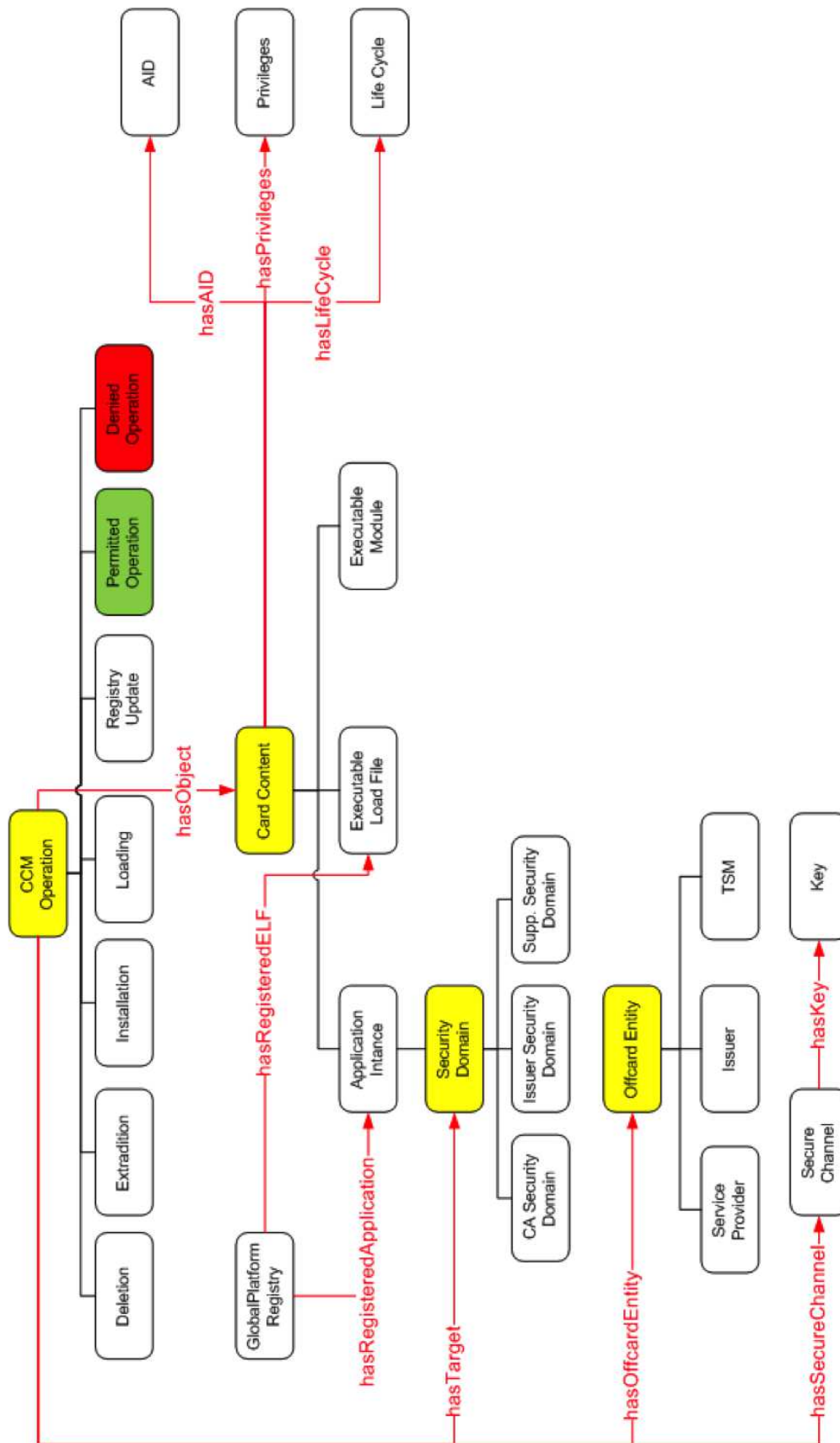


FIGURE 3.14 – L'ontologie GlobalPlatform

Afin d'illustrer l'utilisation de ces règles, nous donnons ci-dessous un exemple de chaque type de règle. La règle suivante vérifie que l'entité extérieure effectuant une opération de gestion de contenu et que le domaine de sécurité cible de cette opération partagent la même clé secrète.

```
@prefix gp: http://www.globalplatform.org/ontologies/
    globalplatform.owl#
@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
@prefix rdfs: http://www.w3.org/2000/01/rdf-schema#

[sameKeyrule: (?operation rdf:type ?o)
(?o rdfs:subClassOf gp:CCMOperation)
(?channel rdf:type gp:SecureChannel)
(?operation gp:hasSecureChannel ?channel)
(?target rdf:type ?t)
(?t rdfs:subClassOf gp:SecurityDomain)
(?key1 rdf:type gp:Key)(?target gp:hasKey ?key1)
(?operation gp:hasTarget ?target)
(?offcard rdf:type ?off)
(?off rdfs:subClassOf gp:OffCardEntity)
(?key2 rdf:type gp:Key)(?offcard gp:hasKey ?key2)
(?operation gp:hasOffCardEntity ?offcard)
(?key1 gp:hasKeyValue ?kv1)(?key2 gp:hasKeyValue ?kv2)
equal(?kv1, ?kv2)
-> (gp:gpSpecificationPolicy gp:permits ?operation)]
```

Dans le domaine de l'implémentation du SE, la règle suivante n'autorise pas la création d'un domaine de sécurité avec le privilège *Delegated Management* car cette fonctionnalité n'est pas supportée.

```
@prefix gp: http://www.globalplatform.org/ontologies/
    globalplatform.owl#
@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
@prefix rdfs: http://www.w3.org/2000/01/rdf-schema#

[unsupportedDMRule: (?operation rdf:type gp:Installation)
(?object rdf:type ?t)
(?t rdfs:subClassOf gp:SecurityDomain)
(?priv rdf:type gp:DelegatedManagementPrivilege)
(?object gp:hasPrivilege ?priv)]
```

```
(?operation gp:hasObject ?object)
-> (gp:gpSpecificationPolicy gp:denies ?operation)]
```

Dans le domaine du champ d'application, la règle suivante autorise la création d'un domaine de sécurité supplémentaire destiné à héberger des applications de paiement, si l'AID du paquetage – plus précisément l'*Executable Load File* enregistré dans le registre GlobalPlatform – a la valeur 'A0 00 00 00 03 53 50' et si l'AID de l'application contenue dans ce paquetage – *Executable Module* – a la valeur 'A0 00 00 00 03 53 50 41' comme spécifié dans [125].

```
@prefix gp: http://www.globalplatform.org/ontologies/
  globalplatform.owl#
@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
@prefix rdfs: http://www.w3.org/2000/01/rdf-schema#
```

```
[sdAIDRule: (?operation rdf:type gp:Installation)
(?em rdf:type ?t)(?t rdfs:subClassOf gp:ExecutableModule)
(?operation gp:hasObject ?em)
(?target rdf:type ?t)(?t rdfs:subClassOf
gp:SupplementarySecurityDomain)
(?operation gp:hasTarget ?target)
(gp:gpGlobalPlatformRegistry
gp:hasRegisteredExecutableModule ?object)
(?elf rdf:type gp:ExecutableLoadFile)
(?elf gp:hasExecutableModule ?em)
(?aid1 rdf:type gp:AID)(?em gp:hasAID ?aid1)
(?aid1 gp:hasAIDValue ?av1)
equal(?av1, "A000000151535041")
(?aid2 rdf:type gp:AID)(?elf gp:hasAID ?aid2)
(?aid2 gp:hasAIDValue ?av2)
equal(?av2, "A0000001515350")
-> (gp:gpSpecificationPolicy gp:permits ?operation)]
```

3.4.5.2 La couche applicative : peupler l'ontologie et raisonner

Pour créer tous les individus nécessaires dans l'ontologie GlobalPlatform, nous faisons appel au gestionnaire d'individus afin de créer les classes RDFS et les triplets RDF. Une fois l'ontologie peuplée, nous faisons appel aux raisonneurs afin de classer les individus. Deux classes sont alors utilisées pour la classification d'une opération de

gestion de contenu : *DeniedOperation* et *PermittedOperation*, toutes deux sous-classes de *CCMOperation*. La classe *CCMOperation* hérite elle-même de la classe *Action*.

La classification est régie par les règles injectées dans le moteur d'inférence. La dernière ligne des règles (par exemple -> (`gp:gpSpecificationPolicy gp:permits ?operation`)) définit comment l'opération doit être classifiée. Par exemple, la règle `unsupportedDMrule` définit que si un individu de type *SecurityDomain* associé à l'individu de type *CCMOperation* a le privilège *DelegatedManagement*, alors l'opération est interdite. La décision est dictée par le prédicat *permits* ou *denies*. En effet, l'individu *gpSpecificationPolicy* a une propriété d'objet *permits* dont le domaine est *PermittedOperation*. Ainsi, l'individu *CCMOperation* est classifié comme *PermittedOperation*.

3.4.5.3 La couche de décision : prendre les décisions et s'interfacer avec l'utilisateur

La couche de décision est l'interface entre l'utilisateur et le framework. C'est une API exposant les fonctionnalités de la couche applicative. Elle est composée de classes permettant d'envoyer des requêtes au modèle afin d'effectuer des opérations de gestion de contenu. Ces classes sont respectivement *InstallationRequest*, *LoadingRequest*, *DeleteRequest*, *ExtraditionRequest* et *RegistryUpdateRequest*. Elles héritent de la classe *CCMOperationRequest*. Afin de bien comprendre comment ces classes fonctionnent et sont utilisées, étudions la classe *LoadingRequest*. Le constructeur de cette classe prend en paramètre les éléments suivants :

- le paquetage à charger (*Executable Load File*, contenant un ou plusieurs *Executable Module*, un AID a été assigné à chacun de ces composants) ;
- l'entité extérieure exécutant l'opération de chargement (*Offcard Entity*) ;
- optionnellement, un bloc DAP (*Data Authentication Pattern*). Le bloc DAP est un mécanisme permettant de s'assurer de l'intégrité du code chargé ;
- optionnellement un jeton de *Delegated Management*. Le jeton de *Delegated Management* est un mécanisme permettant de s'assurer que l'opération effectuée par délégation a été préalablement autorisée par l'émetteur du SE.

Une fois l'objet *LoadingRequest* créé, il est envoyé au framework qui crée les individus correspondants dans l'ontologie. Ensuite, les raisonneurs classifient les individus et le gestionnaire de décision déduira si l'opération est autorisée ou non. Les raisons d'interdire l'opération sont multiples :

- l'entité extérieure et le domaine de sécurité cible ne partagent pas la même clé secrète ;

- le paquetage ou les applications qu’il contient sont déjà présents sur le SE ;
- la requête ne contient pas de jeton de *Delegated Management* alors qu’elle le devrait ;
- la requête ne contient pas de bloc DAP alors qu’elle le devrait.

Si l’opération est rejetée, alors les individus créés dans l’ontologie sont supprimés et l’ontologie est restaurée à son état initial. Si au contraire l’opération est permise, alors les paquetages et les applications contenues sont enregistrés dans le registre GlobalPlatform. Ainsi, il est possible de modéliser un SE particulier en créant une opération de chargement pour tous les paquetages qui doivent être présents et une opération d’installation pour tous les domaines de sécurité et instances d’application qui doivent également se trouver sur le SE.

A cet instant, on peut caractériser un SE donné par les données suivantes que nous appelons « configuration » :

- la différence entre l’ontologie enrichie et l’ontologie initiale. Cette différence consiste en une collections d’objets OWL ;
- les règles définissant l’implémentation du SE ;
- les règles régissant le domaine d’application ;
- optionnellement, les règles décrivant la spécification GlobalPlatform dans le cas où le SE implémente une version différente de celle du modèle.

Une fois ces éléments générés, nous pouvons les réinjecter dans le framework et ainsi obtenir le modèle *ad hoc* complet du SE ciblé. Cela signifie que nous avons rendu la configuration d’un SE *sérialisable* et *désérialisable*. Ces propriétés s’avèrent extrêmement intéressantes dans le contexte du déploiement des services mobiles sans contact. En effet, si un SE embarque dans sa mémoire la configuration qui le caractérise, alors un TSM peut récupérer ces informations et s’en aider pour connaître la marche à suivre afin de déployer un service donné. Cette phase d’aide au déploiement est détaillée dans la section suivante.

3.4.6 Application à la conception d’un outil pour l’aide au déploiement

Comme nous l’avons vu dans l’introduction de ce chapitre, les émetteurs de carte à puce des domaines bancaires et télécommunications utilisent les CMS pour connaître la configuration de leurs cartes une fois déployées sur le terrain. Dans le contexte du NFC, cette gestion se complexifie car le SE, contrairement à la carte à puce plastique, est un environnement dynamique. En effet, le SE n’est pas bloqué par

l'émetteur afin de laisser la possibilité d'y charger des services mobiles sans contact tout au long de son cycle de vie. On aperçoit ici alors les limites des CMS dans le contexte du NFC. Aujourd'hui, les pilotes NFC sont effectués sur des échantillons d'individus réduits, avec un seul émetteur, un seul TSM et un seul SE. Mais lors de déploiements massifs, les CMS montreront leurs limites notamment dans la résolution des problèmes suivants :

- gestion du SE par plusieurs TSMs ;
- gestion de plusieurs SE dans le même équipement ;
- généricité de la gestion du SE, quel que soit son facteur de forme.

En tirant partie de la réflexivité de notre modélisation, nous proposons dans cette section une solution afin de palier aux limites des approches basées sur les CMS.

Comme expliqué à la section précédente, notre modélisation présente les fonctionnalités de réflexivité et de sérialisation/désérialisation. La réflexivité est permise grâce au moteur de requête intégré à Jena basé sur SPARQL. Sur les quatre types de requête proposés, nous n'utilisons que le *SELECT* et le *ASK*. Les requêtes non utilisées, *CONSTRUCT* et *DESCRIBE*, retournent un graphe RDF et ne sont pas adaptées à notre besoin. *SELECT* nous permet d'extraire des données tandis que *ASK* nous permet de tester l'existence d'un individu dans l'ontologie. Nous illustrons ces concepts par l'étude d'un cas concret : un TSM doit déployer l'application de paiement Visa sur un SE dont il n'a pas de connaissance préalable. Voici les étapes suivies par le TSM, illustrées à la figure 3.15 :

1. Le TSM se connecte au SE et en extrait les données de configuration ;
2. Le TSM injecte ces données de configuration dans le framework GlobalPlatform ;
3. Le TSM, via le framework, envoie une requête de génération de scripts de déploiement pour l'application Visa
4. Le framework crée alors un objet d'une nouvelle classe – *DeploymentScenarioRequest* – contenant les informations suivantes :
 - AID du paquetage ('A0 00 00 00 03 12')
 - AID de l'application contenue dans le paquetage ('A0 00 00 00 03 12 56')
 - AID de l'instance d'application Visa à créer ('A0 00 00 00 03 10 10')
 - domaine de sécurité cible
5. Le framework envoie une requête de type *ASK* pour déterminer si un paquetage avec l'AID 'A0 00 00 00 03 12' est déjà présent sur le SE. Si non, le framework ajoute au script de déploiement les commandes correspondant au chargement du paquetage Visa ;
6. Si oui, le framework envoie une requête de type *ASK* pour déterminer si une application avec l'AID 'A0 00 00 00 03 12 56' est associée au paquetage issu

de la requête précédente. Sinon, le paquetage est supprimé et remplacé par le paquetage adéquat en ajoutant au script de déploiement les commandes de chargement ;

7. Si oui, le framework envoie une requête de type *SELECT* pour obtenir la liste de toutes les applications Visa déjà instanciées sur le SE, *i.e.* toutes les instances d'application dont l'AID commence par 'A0 00 00 00 03 10 10' ;
8. Si aucune instance de l'application Visa n'est déjà présente, une instance avec l'AID 'A0 00 00 00 03 10 10' sera créée. Si une ou des instances de l'application Visa existent déjà, le framework décide de l'AID à utiliser pour cette instance (par exemple 'A0 00 00 00 03 10 10 xy', xy étant le numéro de l'instance). Le framework ajoute alors au script de déploiement les commandes correspondant à l'instantiation d'une application Visa avec son AID.
9. Le framework envoie une requête de type *SELECT* afin de recueillir des informations sur le domaine de sécurité cible comme le mode de gestion et l'identifiant de sa clé ;
10. En fonction de ces informations, le script est envoyé à un module afin d'être mis en conformité avec le niveau de sécurité, le mode de gestion et la clé utilisée par le domaine de sécurité ;
11. Le TSM récupère le script de déploiement et le transmet tel quel, commande par commande, au SE.
12. Le TSM récupère également les données de configuration du SE mis à jour par le framework et les réinjecte dans le SE si l'ensemble des opérations précédentes se sont correctement déroulées.

Nous avons vu dans cette section comment grâce aux propriétés de réflexivité du modèle et de sérialisation/désérialisation des données de configuration du SE, un TSM peut gérer sans CMS, *i.e.* sans connaissance préalable, un SE et y déployer des services mobiles sans contact. Nous voyons en cette approche la solution aux problèmes que poseront les déploiements massifs : gestion par plusieurs TSMs, gestion de plusieurs SE dans un même équipement et généralité de la gestion du SE. Nous venons de démontrer la généralité de la gestion. La gestion par plusieurs TSMs d'un même SE est rendue possible par la localisation des données de configuration. En les installant directement dans le SE, elles sont directement accessibles à tous les TSMs autorisés et remises à jour après chaque déploiement. La gestion de plusieurs SE est rendue possible en injectant de la même manière les données de configuration de tous les SE. Trois localisations pour ces données nous paraissent raisonnables en termes de sécurité : directement sur le SE cible ce qui implique que le TSM ait la connaissance

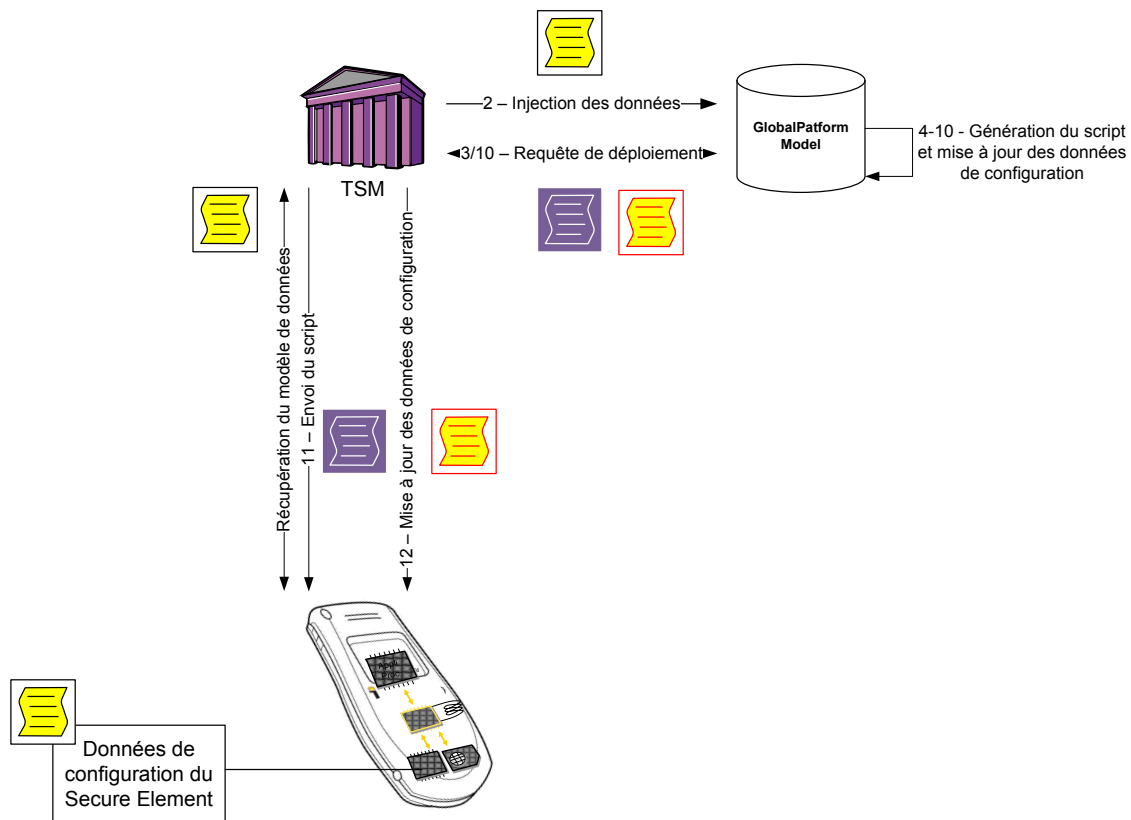


FIGURE 3.15 – Cinématique de déploiement d'un service mobile sans contact

de la cible, sur un SE qui serait défini comme maître et regroupant les données de configuration de tous les SE, mais là encore le TSM doit décider du SE cible, ou bien le choix du SE cible peut être laissé à l'équipement mobile. Dans ce dernier cas, le mobile retourne au TSM les données de configuration du SE choisi. Les données de configuration sont alors stockées dans une partie sécurisée de l'équipement mobile telle que la TEE (*Trusted Execution Environment* définie par GlobalPlatform dans [57]).

Nous exposons dans la section suivante la validation de notre approche.

3.4.7 Validation de l'approche

3.4.7.1 Introduction

Pour valider notre approche, nous avons utilisé deux SE concurrents : le SE NXP et le SE INSIDE Secure, le composant *SecuRead*. Nous avons utilisé le SE NXP sous sa forme de carte à puce plastique (plateforme *JCOP*, Java Card Open Platform) et le SE INSIDE Secure sous forme de banc d'évaluation puis sur d'équipement réel.

Les sections suivantes détaillent la validation sur carte à puce et banc d'évaluation dans un premier temps puis, dans un second temps, sur un équipement réel. Enfin, nous comparons les résultats obtenus sur les différents supports.

3.4.7.2 Validation sur carte à puce NXP JCOP et banc d'évaluation INSIDE SecuRead

Cette section détaille le protocole et les résultats obtenus sur les deux SE concurrents pris de manière isolée, i.e. non intégrés dans un équipement. La carte JCOP utilisée est une JCOP 41 utilisant une puce Philips Semiconductors P8WE5032. Le SE INSIDE Secure utilisé est le SecuRead v1.0 basé sur une puce Infineon.

La différence entre les deux produits réside uniquement dans le temps d'exécution des instructions, nous ne détaillerons que les résultats obtenus avec le banc d'évaluation. Un banc d'évaluation est une carte électronique sur laquelle sont intégrés les composants qui seront ensuite intégrés dans les équipements mobiles. Il permet d'effectuer une série de tests et de validation dans des conditions proches du produit final. Le banc d'évaluation utilisé est représenté à la figure 3.16. Afin de réaliser cette validation, les éléments suivants ont été mis en œuvre :

framework GlobalPlatform : notre framework tel que décrit dans ce chapitre avec une ontologie GlobalPlatform complète ;

données de configuration : nous avons créé des données de configuration sommaires de SecuRead

applet de configuration : nous avons développé une application pour le SE destinée à contenir les données de configuration de ce dernier ;

paquetage de l'application Visa : nous avons créé un paquetage symbolisant l'application Visa. Le cœur de l'application est réduit à son minimum fonctionnel, seuls les AIDs répondent aux pré-requis émis par Visa ;

librairie GlobalPlatform : une librairie développée dans le cadre de cette validation permettant de dialoguer avec un SE, d'accomplir des opérations de gestion de contenu et de gérer la session sécurisée.

La validation est ensuite réalisée suivant les étapes détaillées ci-dessous.

3.4.7.2.1 Récupération des données de configuration du SE

La première étape consiste à récupérer les données de configuration du SE dont nous n'avons aucune connaissance *a priori*. Pour cela, nous avons développé une applet de configuration destinée à stocker ces données. Elle stocke au format ASCII

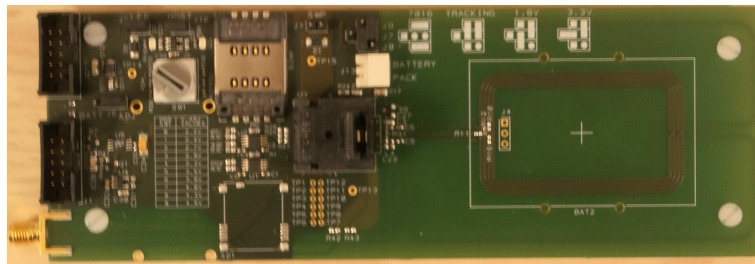


FIGURE 3.16 – Banc d'évaluation utilisé pour la validation

les éléments OWL décrivant le SE, *i.e.* la différence entre l'ontologie GlobalPlatform initiale et l'ontologie enrichie.

L'applet est un simple *File Manager*, *i.e.* une applet permettant de créer des fichiers et d'y écrire/lire des données grâce aux commandes *WRITE/READ BINARY*. Dans notre cas, les données de configuration sont encore vierges comme le montre leur lecture présentée au listing A.1.

3.4.7.2.2 Chargement de l'ontologie et déploiement d'un service mobile sans contact

Ensuite, les données de configuration sont « injectées » dans le framework, *i.e.* on rajoute les éléments OWL des données de configuration à l'ontologie GlobalPlatform et on charge cette ontologie enrichie en mémoire (*cf.* listing A.2). Une requête d'installation d'une application (exemple : l'application de paiement *Paypass Magstripe*, développée dans le cadre de mes activités à INSIDE Secure) est alors créée et envoyée au framework. Comme le montre le listing A.2, le framework crée les individus correspondants, prend les décisions, puis génère le script correspondant au chargement du paquetage et installation de l'application.

3.4.7.2.3 Mise à jour des données de configuration

Une fois le service mobile sans contact déployé sur le SE, les données de configuration doivent être mise à jour. Pour cela, nous sélectionnons l'applet de configuration puis écrasons les données présentes avec la commande *WRITE BINARY* comme le montre le listing A.3.

3.4.7.3 Nouvelle requête de déploiement d'un service mobile sans contact

Cette dernière étape consiste à valider que les données de configuration ont bien été mises à jour, de les charger dans le framework et de réaliser la même requête de déploiement. Comme le montre le listing A.4, les données de configuration récupérées correspondent au contenu applicatif créé lors de l'étape de déploiement

précédente et le framework prend la décision de seulement créer une nouvelle instance de l'application plutôt que de recharger le paquetage entier puisqu'il est présent sur le SE.

3.4.7.4 Etude comparative

Le tableau 3.4 synthétise les temps obtenus sur les deux SE lors de la même opération de requête de déploiement. La première constatation est que la validation a été un succès sur les deux SEs. La solution proposée est donc portable sur les deux principaux SEs du marché. On observe ensuite, de manière logique, que les temps d'initialisation de l'ontologie et de création de la requête sont sensiblement identiques puisque ces opérations ont été réalisées sur le même ordinateur. On observe également que le SE JCOP est légèrement plus rapide en lecture mais bien plus lent en écriture. Cela a un impact non négligeable puisqu'au final la même opération prend deux fois plus de temps sur le SE JCOP 41 que sur le SE INSIDE Secure SecuRead.

	<i>SecuRead</i>	<i>JCOP</i>
Lecture de la configuration depuis l'applet	47ms	47ms
Initialisation de l'ontologie	1797ms	1890ms
Création de la requête et génération des scripts	203ms	203ms
Envoi du 1 ^{er} script au SE	6000ms	4859ms
Ecriture de la configuration dans l'applet	10188ms	28453ms
Lecture de la configuration depuis l'applet	563ms	344ms
Initialisation de l'ontologie enrichie	469ms	485ms
Création de la 2 ^{nde} requête et génération des scripts	47ms	47ms
Envoi du 2 nd script au SE	1109ms	1141ms
Temps total	20937ms	38484ms

TABLE 3.4: Comparaison des temps d'exécution des différentes phases

3.4.7.5 Validation sur un équipement réel

L'équipement réel choisi est un téléphone mobile BlackBerry Bold 9900 embarquant le composant SecuRead. La gestion d'un tel SE est un peu particulière. En effet, contrairement à la carte SIM (UICC) qui peut être administrée à distance par le biais de SMS par exemple, le SE embarqué ne dispose pas de ce type de mécanisme. Un composant supplémentaire doit être installé sur le mobile afin de réaliser le routage entre le TSM et le SE. On retrouve ce composant dans la littérature sous la dénomination de *TSM Proxy MIDlet* ou encore de *Admin Agent* dans les standards GlobalPlatform [67]. Afin de valider le déploiement de services mobiles sans contact

sur un SE embarqué, nous avons développé une plateforme de tests appelée « TSM Lab ».

3.4.7.5.1 TSM Lab

Le TSM Lab a deux vocations : permettre la validation de notre approche sur un équipement réel et apporter à INSIDE Secure la connaissance de ce maillon de la chaîne et ainsi, la possibilité de tester la gestion OTA de son portfolio de services mobiles sans contact. La cinématique d'utilisation du TSM Lab est la suivante :

1. L'utilisateur se rend sur une page web sur laquelle il saisie l'adresse email associée à son BlackBerry ;
2. L'utilisateur sélectionne, parmi ceux proposés, les services mobiles sans contact qu'il souhaite installer ;
3. Un email contenant les informations de connexion est envoyé sur le BlackBerry. Cet email est alors intercepté par un module dédié de la *TSM Proxy MIDlet*. Elle se connecte alors au serveur afin d'initier une session d'administration à distance.
4. Le serveur envoie les scripts de déploiement au SE via la MIDlet.

L'architecture du TSM Lab et la cinématique d'utilisation sont présentées à la figure 3.17. Dans la section suivante, nous exposons comment le TSM Lab a été utilisé pour la validation de notre approche sur un équipement réel.

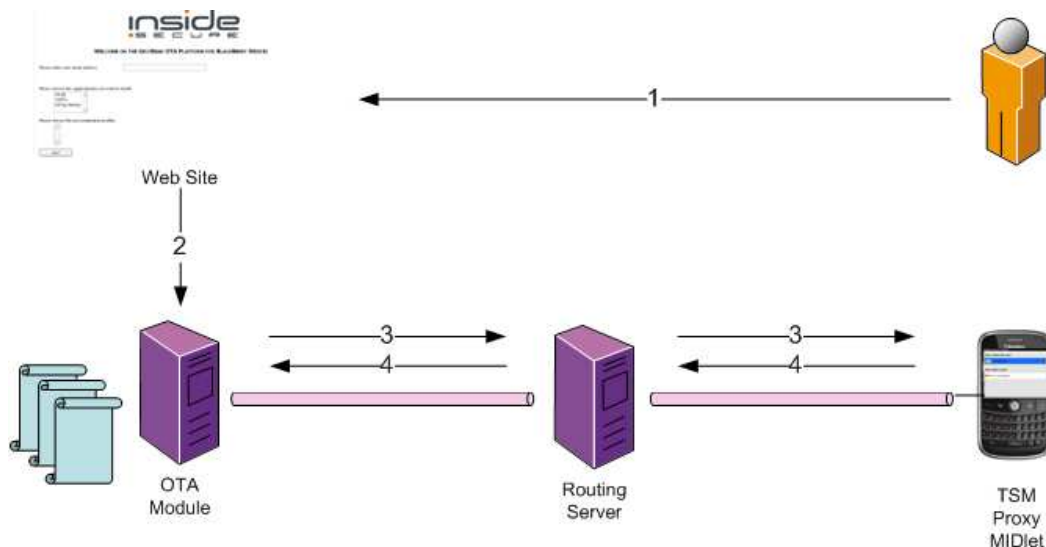


FIGURE 3.17 – Architecture et cinématique d'utilisation du TSM Lab

3.4.7.6 Validation

Afin de réaliser la validation de notre approche sur un équipement réel, les éléments suivants ont été mis en œuvre :

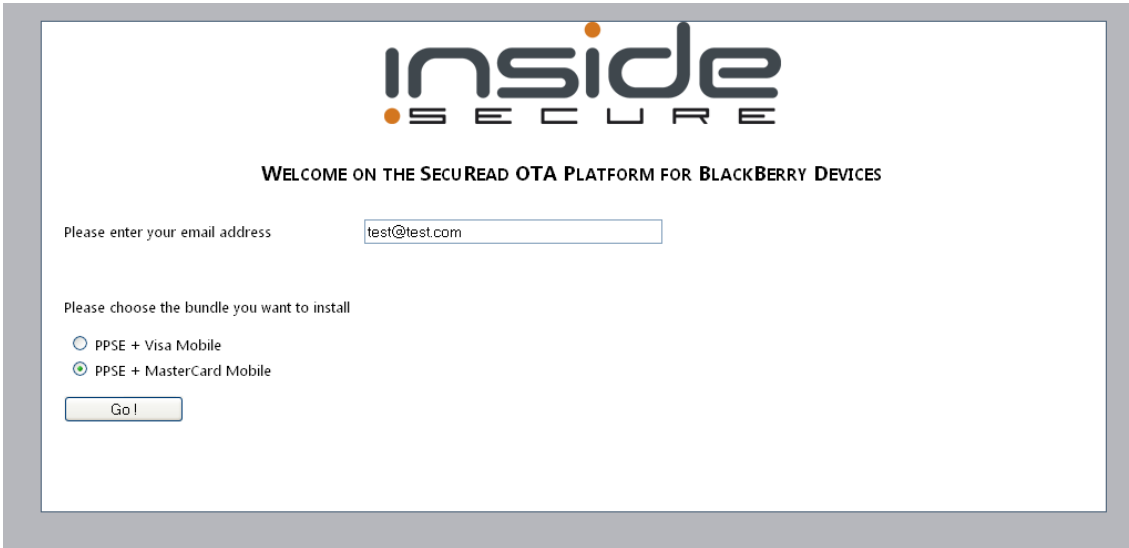
framework GlobalPlatform

données de configuration

paquetage de l'application Visa

TSM Lab : le TSM Lab, décrit précédemment, a été quelque peu modifié afin de procéder à la validation.

Après avoir sélectionné sur le site web le service mobile que l'utilisateur souhaite installer comme illustré à la figure 3.18, un email est reçu par l'équipement cible et provoque le lancement de l'application TSM. Lors de cette validation, l'application proxy TSM a été intégrée à une application déjà existante appelée « Wallet ». Le Wallet est une application représentant les différentes applications de paiement présentes sur le SE et permettant à l'utilisateur d'effectuer des opérations telles que la suppression de cartes « virtuelles » ou la réorganisation de leurs priorités respectives vis-à-vis du terminal de paiement. Ensuite, le Wallet se connecte au serveur du TSM Lab et celui-ci récupère les données de configuration en interrogeant l'applet de configuration. Ces deux opérations s'exécutent de manière invisible à l'utilisateur. Via le framework GlobalPlatform, le TSM Lab génère ensuite le script de déploiement du service mobile sélectionné par l'utilisateur. Enfin, le TSM Lab envoie le script de déploiement au SE via le Wallet. Cette opération est matérialisée par l'apparition d'une barre de progression puis l'apparition d'une carte virtuelle correspondant à l'application sélectionnée comme illustré à la figure 3.19. La dernière consiste à l'écriture des données de configuration dans l'applet de configuration, encore une fois, de manière invisible pour l'utilisateur.



inside
SECURE

WELCOME ON THE SECUREAD OTA PLATFORM FOR BLACKBERRY DEVICES

Please enter your email address

Please choose the bundle you want to install

PPSE + Visa Mobile

PPSE + MasterCard Mobile

FIGURE 3.18 – Site web du TSM Lab



FIGURE 3.19 – Étapes de la validation sur un équipement réel : (a) « réveil » du Wallet, (b) réception du script de déploiement, (c) instantiation du service mobile et de la carte virtuelle correspondante

3.4.8 Discussion

La modélisation d'un SE GlobalPlatform par les ontologies nous a permis d'implémenter une plateforme de génération automatique de scripts de déploiement. Elle répond aux nouveaux besoins liés au déploiement des services mobiles sans contact sur un parc de SEs volumineux et surtout hétérogène. Les membres du consortium GlobalPlatform nous ont fait part de leur intérêt dans la modélisation proposée car ils la considèrent comme une amélioration conséquente de l'existant, *i.e.* les profils système GlobalPlatform datant de 2003.

3.5 Conclusion

Dans ce chapitre, nous avons proposé trois contributions au déploiement des services mobiles sans contact : une modélisation d'un Secure Element, un mécanisme de distribution des clés à la demande et un terminal de paiement électronique sur téléphone mobile.

Le mécanisme de distribution des clés d'un Secure Element à la demande est une contribution d'anticipation. Alors que les organismes de standardisation se concentrent actuellement sur le téléphone mobile géré de manière classique par un TSM, nous nous sommes intéressés aux cas d'usages futurs. Ces cas d'usages concernent les équipements de type ordinateur personnel, tablette, ... avec lesquels le schéma d'émission diffère des schémas traditionnels. Prenons l'exemple d'un fabricant de tablettes qui souhaite intégrer la technologie NFC à ses équipements afin d'offrir des services innovants à ses utilisateurs. Nous considérons qu'il est un frein à l'adoption massive du NFC que ces fabricants soient contraints de gérer un système de gestion des clés ou de lourdes relations techniques avec les TSMs. En cela, notre contribution leur offre la possibilité de déléguer cette gestion des clés à une entité tierce de confiance qui se chargera de leur distribution auprès des TSMs accrédités au moment de la « prise de possession » du Secure Element. Ayant reçu un écho extrêmement favorable de la part de ses partenaires, INSIDE Secure a décidé de standardiser ce mécanisme. Nous avons présenté, argumenté et débattu cette contribution auprès de la *Mobile Task Force* de GlobalPlatform et participons actuellement à sa mise en forme au sein du *Card Specification Working Group*.

Nous avons proposé trois architectures pour transformer un téléphone mobile NFC en un terminal de paiement mobile sans contact. Elles nécessitent toutes les trois un effort d'intégration par les fabricants de mobiles mais offrent toutes les trois un haut niveau de sécurité. Parmi ces architectures, celle combinant la TEE et le Secure Element se démarque et semble la plus viable à long terme. Dans cette

configuration, la TEE héberge les applications de paiement terminal et le Secure Element est utilisé comme un SAM logiciel. La TEE est également grandement mise à contribution pour ses capacités à afficher le montant de la transaction et à recevoir un code PIN depuis le clavier ou l'écran de manières sécurisées. Nous avons également spécifié une solution permettant de se conformer totalement aux standards de sécurité PCI en termes d'acceptations de paiements mobiles. Cependant, cette solution ne peut être exposée dans ce mémoire pour des raisons de confidentialité.

Enfin, la modélisation d'un Secure Element GlobalPlatform par les ontologies permet de s'affranchir des systèmes de gestion des cartes (CMS). En effet, la description du Secure Element n'est plus localisée sur un serveur, dans une base de données, mais directement sur le Secure Element. Une fois récupérée par le TSM, elle est injectée dans le modèle. Celui-ci est doté de capacités de réflexivité et de raisonnement, qui lui permettent de générer automatiquement les scripts de déploiement de services mobile sans contact en fonction du contenu applicatif existant et de la configuration du Secure Element. Comme nous l'avons évoqué à la section 3.4.2.7, nous nous sommes efforcé d'encapsuler la modélisation proposée par GlobalPlatform en 2003. Cette implémentation consistait en une description statique de profils pour effectuer des opérations de gestion de contenu. Ces profils étaient accompagnés d'un langage de scripting permettant d'interpréter et d'exécuter ces profils. Non seulement nous reprenons les principaux concepts exprimés dans ces profils, mais nous les améliorons en apportant une modélisation à deux dimensions (statique et dynamique) et incluons de manière native et performante la génération automatique des scripts. Ces travaux ont fait l'objet d'un article accepté et présenté à la conférence *NFC Workshop 2012* [126] et d'une présentation à *WIMA NFC Monaco*.

Nous avons proposé dans ce chapitre des contributions pour faciliter le déploiement et l'acceptation des services mobiles sans contact. Dans le chapitre suivant, nous nous intéressons à la phase d'utilisation d'un service extrêmement sensible : le paiement. En effet, le paiement est un service qui véhicule des données personnelles et financières, ses transactions doivent donc assurer un niveau de sécurité adéquat. Dans le chapitre suivant, nous proposons une plateforme logicielle destinée à l'analyse de la sécurité des transactions de paiement.

Chapitre 4

Plateforme d'analyse de la sécurité des transactions de paiement

Ce chapitre présente une plateforme d'analyse des transactions permettant de reproduire de manière logicielle, les attaques de l'état de l'art concernant le paiement EMV. Nous présentons également un framework de fuzzing pour carte à puce développé sur cette base et permettant de réaliser des tests fonctionnels et sécuritaires sur des services mobiles.

Sommaire

4.1	Introduction	96
4.2	Le protocole EMV	96
4.3	Etat de l'art	103
4.4	Le framework « WinSCard Tools »	109
4.5	Un framework de fuzzing pour les services de paiement mobile .	123
4.6	Conclusion	153

Présentation

QUELLES sont les attaques auxquelles les transactions de paiement sont sensibles ? Comment les reproduire et s'en protéger ? Ce chapitre répond à ces questions en présentant une plateforme d'analyse des transactions de paiement permettant de reproduire les attaques de l'état de l'art et de proposer l'amélioration d'une méthode de test basée sur le fuzzing.

Mots clés :

EMV, sécurité, transaction de paiement, fuzzing

Contributions de ce chapitre

- La mise en application de l'outil *WinSCard Tools* développé au laboratoire GREYC
- La reproduction logicielle des attaques majeures de l'état de l'art.
- Une méthodologie pour améliorer les tests par fuzzing sur les applications de paiement.

4.1 Introduction

L'utilisation d'un service, quel qu'il soit, est la dernière phase de son cycle de vie avant sa désactivation, c'est aussi la plus sensible. En effet, durant cette phase, le service est exposé aux utilisateurs finaux et peut être la cible d'attaques. Dans ce chapitre, nous nous intéressons au paiement EMV, un des services les plus sensibles puisqu'il fait appel à des données personnelles et financières. Nous exposons nos contributions à l'outil *WinSCard Tools*, outil d'analyse des transactions, afin d'implémenter des attaques de l'état de l'art et une proposition de méthodologie pour les tests par fuzzing.

Organisation du chapitre

Les sections suivantes présentent nos contributions. La section 4.2 présente le protocole de paiement EMV. La section 4.3 présente un état de l'art des méthodes de tests de pénétration, des attaques sur les cartes à puce et des plateformes d'analyse de transactions. La section 4.4 présente l'outil *WinSCard Tools* développé au laboratoire GREYC et son application à l'analyse des transactions. La section 4.5.1 présente une méthodologie améliorant les tests des applications de paiement par fuzzing.

4.2 Le protocole EMV

Le protocole EMV définit une transaction de paiement, basée sur l'utilisation de la carte à puce, sécurisée et interopérable à l'international. Les sections suivantes abordent EMV sous deux angles majeurs, *i.e.* les aspects transactionnels et sécuritaires du protocole.

4.2.0.1 Aspect transactionnel

Une transaction EMV se déroule en 14 étapes dont 10 obligatoires (Figure 4.1). Elle se découpe en 3 grandes phases : initialisation, analyse de la transaction et décision. Durant la phase d'initialisation et d'analyse de la transaction le terminal et la carte réalisent des contrôles dont ils inscrivent le résultat respectivement dans le TVR (*Terminal Verification Results*) et le CVR (*Card Verification Results*). Ces éléments sont utiles au terminal pour la 3ème phase, la prise de décision.

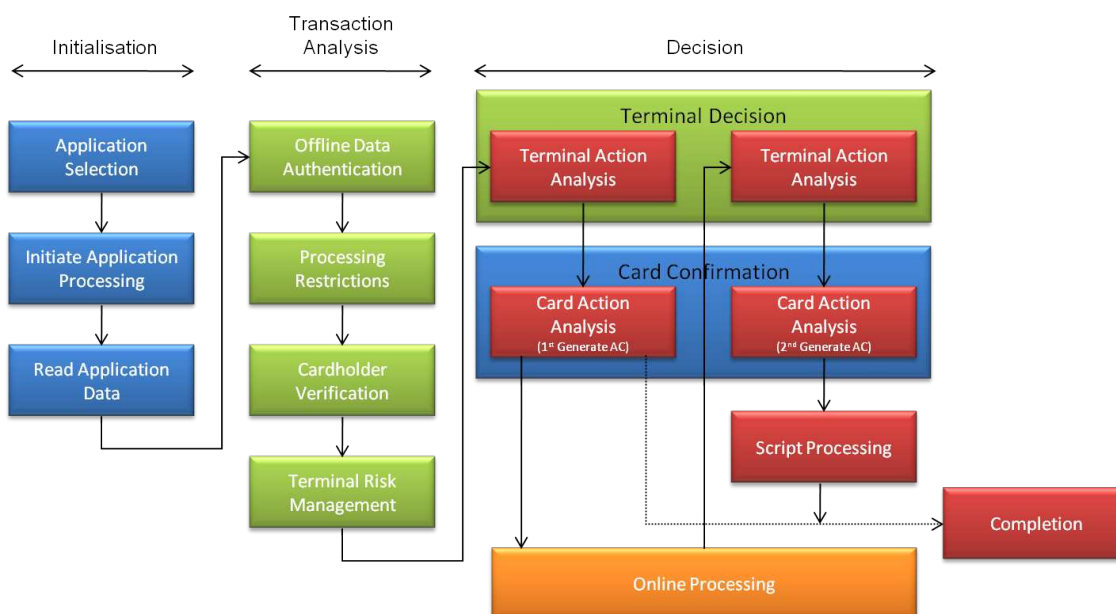


FIGURE 4.1 – Cinématique d'une transaction EMV

4.2.0.1.1 Initialisation de la transaction

Durant cette phase, le terminal a pour objectif de :

- définir puis sélectionner l'application carte à utiliser pour mener la transaction,
- d'initier la transaction de paiement,
- de lire depuis la carte les données dont il a besoin pour mener la transaction.

Ces étapes se réalisent par l'envoi des commandes successives :

- SELECT [PSE] et/ou SELECT [applications connues par le terminal]¹,

1. Le PSE (*Payment System Environment*) est une application qui référence les applications de paiement présentes sur la carte. Sa présence n'est pas obligatoire, s'il est absent le terminal sélectionne les applications de paiement dont il possède l'application côté terminal et établit une liste d'applications candidates.

- SELECT [application choisie], pour sélectionner l'application de paiement choisie lors de la précédente étape ;
- GET PROCESSING OPTIONS, afin d'initier la transaction et de permettre à la carte d'informer le terminal des données dont elle a besoin pour la transaction ;
- une série de READ RECORD successifs, pour lire les infos dites "publiques" de la carte et nécessaires pour effectuer la transaction.

4.2.0.1.2 Analyse de la transaction

Pendant la phase d'analyse de la transaction, le terminal effectue les opérations suivantes :

- la vérification de l'intégrité de la carte et de ses données,
- la vérification qu'il n'existe aucune restriction pour la transaction,
- la vérification du porteur,
- la gestion du risque côté terminal.

La vérification de l'intégrité de la carte et de ses données se fait soit de manière statique (SDA, *Static Data Authentication*) ou dynamique (DDA, *Dynamic Data Authentication*). Cette opération consiste à récupérer, par un jeu de vérification de certificats contenus dans la carte, la clé publique utilisée pour signer les données et à vérifier la signature de ces dernières. Lorsque cette vérification est réalisée de manière statique (SDA), la carte peut alors être clonée très facilement. Pour corriger cela, l'introduction d'un aléa rend cette vérification dynamique, la carte non clonable et la transaction non rejouable. Pour ce faire, le terminal envoie des données aléatoires et variant dans le temps à la carte qu'elle utilisera pour la génération de la signature des données.

La vérification des restrictions consiste à vérifier que l'application carte est activée, n'est pas expirée, est compatible avec le type de transaction en cours et que les applications carte et terminal sont compatibles, etc.

La vérification du porteur est une étape importante et qui nous intéresse tout particulièrement dans cette étude. Lors d'une transaction EMV, le terminal est en contact permanent avec la carte. Ainsi, le terminal a un contrôle total sur la transaction et peut notamment l'annuler en cas d'arrachement de la carte. Cela lui donne également la possibilité de mettre la transaction en "pause" pour procéder à la vérification du porteur puis, si cette dernière a réussi, continuer la transaction. EMV définit 5 méthodes de vérification du porteur : pas de vérification (c'est une méthode définie!), la vérification de la signature du porteur, la vérification du code

PIN (*Personal Identification Number*) par la carte – en clair et chiffrée – et la vérification du code PIN chiffré *online* (par le serveur d'autorisation bancaire). Les méthodes acceptées par la carte sont indiquées dans le *CVM List* détenu par la carte. Lorsque qu'une méthode échoue, le terminal essaie la suivante jusqu'au succès de la vérification du porteur ou le rejet de la transaction.

N.B : il est important de noter que certains réseaux bancaires comme ceux de la France, l'Italie ou encore l'Allemagne n'acceptent la vérification du PIN *online* que pour les opérations de retrait sur les distributeurs.

La gestion du risque côté terminal consiste essentiellement en 2 contrôles : le contrôle du montant de la transaction – s'assurer que le montant de la transaction (et le montant cumulé des transactions précédentes, si l'information est disponible) ne dépasse pas un certain seuil – et le contrôle du compteur du nombre de transactions consécutives réalisées offline – s'assurer qu'il ne dépasse pas un certain nombre.

4.2.0.1.3 Décision

La phase de décision se décompose en :

- décision du terminal,
- confirmation par la carte,
- la demande d'autorisation (optionnel),
- l'envoi de scripts par l'émetteur (optionnel).

La prise de décision par le terminal se fait en effectuant une comparaison de 3 éléments : le TVR, les IAC (*Issuer Action Code*) et les TAC (*Terminal Action Code*). Les IAC sont lus depuis la carte (lors de la phase d'initialisation) et reflètent l'action souhaitée par l'émetteur basée sur les résultats des contrôles indiqués dans le TVR. Les TAC sont présents dans le terminal et reflètent de la même manière les actions souhaitées par l'acquéreur (banque du commerçant). Il existe 3 types d'*Action Code* : *Denial*, *Online* et *Default*, soient 3 IAC et 3 TAC. Chacun est comparé bit à bit avec le TVR. Si pour un bit du TVR à 1, on retrouve à la même position un bit à 1 dans l'IAC ou le TAC la décision prise est celle indiquée par le type de l'*Action Code* (ex. : si le TVR et l'IAC *Denial* ont tous les deux un bit à 1 à la même position, le terminal prend la décision de refuser la transaction).

Le terminal informe la carte de la décision prise par le biais de la commande GENERATE APPLICATION CRYPTOGRAM. A la réception de la commande GENERATE AC, la carte procède également à des contrôles pour confirmer ou aller contre la décision du terminal. En règle générale, la carte suit les règles suivantes :

- la carte confirme la décision du terminal,

- si le terminal veut approuver la transaction *offline*, la carte peut décliner la transaction *offline* ou décider d'aller *online*,
- si le terminal souhaite réaliser la transaction *online*, la carte peut la décliner *offline*

Quelle que soit la décision prise par la carte, celle-ci génère un cryptogramme. Celui-ci est envoyé au terminal lors de la réponse à la commande GENERATE AC. Le type de cryptogramme, *i.e.* la décision prise par la carte est renseignée dans l'élément *Cryptogram Information Data* (CID).

La décision prise par la carte est guidée par la phase de gestion du risque par la carte. Lors de cette phase, la carte vérifie, entre autres, les compteurs *offline*. Ces compteurs indiquent le nombre de transactions consécutives réalisées *offline* et le montant cumulé des transactions réalisées *offline*. Si l'un de ces deux compteurs atteint sa limite, la carte forcera la transaction à être *online* ou décliner la transaction en fonction des paramètres de gestion de risque fixés par l'émetteur.

Lorsqu'une transaction doit être autorisée *online*, *i.e.* par le serveur d'autorisation bancaire, il est à la charge du terminal de construire le message d'autorisation et de l'envoyer sur le réseau. Ce message est construit à partir de divers éléments provenant à la fois de la carte et du terminal comme l'*Application Interchange Profile* (AIP), l'*Application Transaction Counter* (ATC), l'*Authorisation Request Cryptogram* (ARQC, cryptogramme généré par la carte), le TVR, un aléa, ...

Selon un algorithme à la discrétion de l'émetteur, la transaction est autorisée ou non et la réponse est envoyée au terminal. Le terminal analyse le code réponse puis renvoie une nouvelle commande GENERATE AC à la carte – optionnellement, l'émetteur peut s'authentifier auprès de la carte. C'est lors de la réception de la demande d'autorisation que la carte remet à zéro les compteurs.

Une dernière phase, optionnelle, et qui n'a lieu que lorsque la transaction est *online* est le script processing. Ce mécanisme offre la possibilité aux émetteurs de gérer à distance leurs cartes. L'émetteur peut ainsi envoyer des commandes pour mettre à jour des données ou débloquer le code PIN par exemple.

4.2.0.2 Aspects sécuritaires

4.2.0.2.1 Authentification de la carte

L'authentification *offline* de la carte peut s'effectuer soit de manière statique soit de manière dynamique. L'authentification statique SDA met en jeu les données de personnalisation de la carte afin de s'assurer que la carte présentée n'a subi aucune

altération depuis son émission. Au moment de la personnalisation par l'émetteur (la banque), celui-ci « injecte » dans la carte des données en clair et une signature d'une partie de ses données. Il injecte également le certificat émetteur, *i.e.* sa clé publique chiffrée par la clé privée de l'autorité de certification (Visa, MasterCard, JCB, American Express ...) comme illustré à la figure 4.2.

Lors d'une transaction de paiement, le terminal récupère le certificat émetteur, il le vérifie grâce à la clé publique de l'autorité de certification qu'il stocke dans sa mémoire sécurisée, en récupérant la clé publique de l'émetteur qu'il utilise alors pour vérifier que les données signées correspondent à la signature d'une partie des données de personnalisation (pointées par l'*Application File Locator* (AFL)).

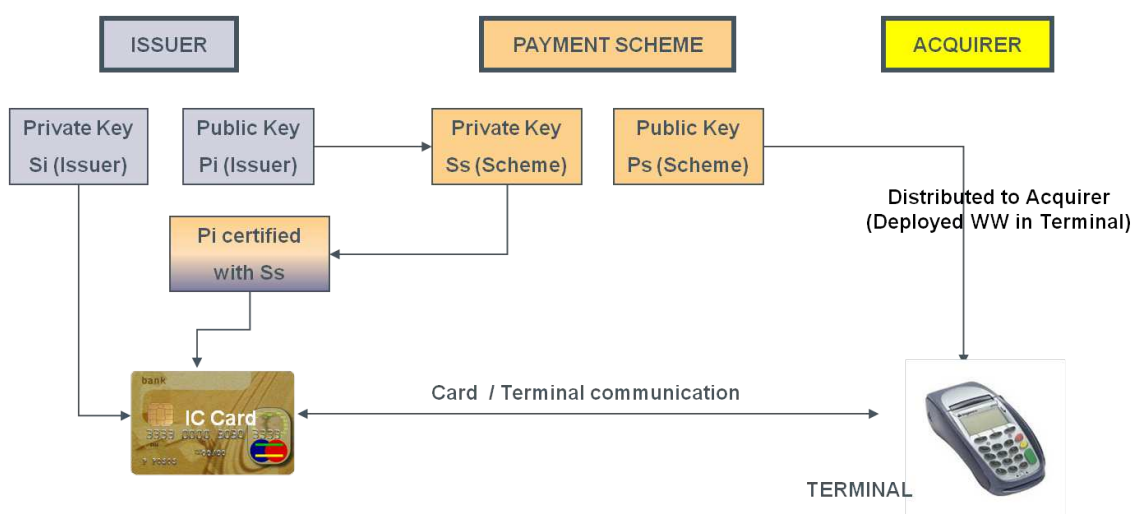


FIGURE 4.2 – Static Data Authentication (Source : INSIDE Secure)

L'authentification dynamique consiste à générer une signature des données à chaque transaction et ce, en s'appuyant sur un aléa transmis par le terminal. Ainsi, le terminal est un élément actif de l'authentification. Pour effectuer cette génération dynamique de signature, la carte doit être dotée de capacités cryptographiques. Elle est personnalisée avec une clé privée, un certificat carte (clé publique de la carte chiffrée avec la clé privée de la banque) et comme pour SDA avec le certificat émetteur comme illustré à la figure 4.3. Le terminal envoie la commande `INTERNAL AUTHENTICATE` à la carte avec un aléa. Cette dernière génère alors une signature unique des données indiquées dans l'AFL et de l'aléa envoyé par le terminal.

Le mécanisme de DDA peut être combiné à la réponse de la commande `GENERATE AC`. Cette méthode appelée *Combined DDA/Application Cryptogram Generation* (CDA) garantit en générant une signature dynamique que les données de la réponse n'ont pas été altérées. En effet, ces données sont particulièrement sensibles puisqu'elles contiennent la décision de la carte sur la transaction en cours. Lorsque

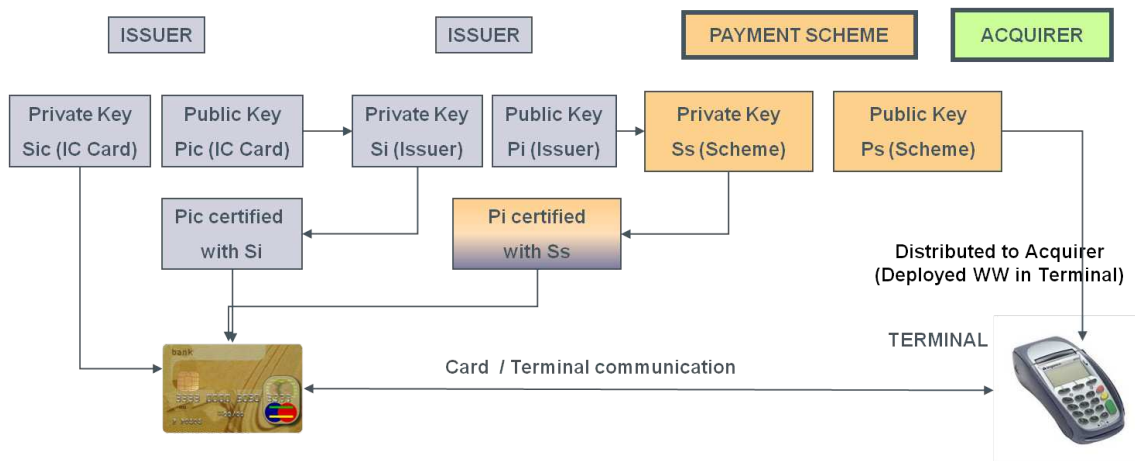


FIGURE 4.3 – Dynamic Data Authentication (Source : INSIDE Secure)

ces données ne sont pas protégées, alors une attaque *man-in-the-middle* pourrait modifier un refus ou une demande de transaction *online* de la part de la carte en une acceptation *offline* aux yeux du terminal.

4.2.0.2.2 Authentification du porteur

EMV supporte cinq méthodes d'authentification du porteur :

- PIN chiffré *online*,
- PIN chiffré *offline*,
- PIN en clair *offline*,
- signature,
- pas de vérification.

Les options que souhaite supporter l'émetteur pour l'authentification de son porteur sont exprimées sous la forme d'une *Cardholder Verification Method List* (CVM List). Cette liste contient les méthodes supportées, la condition dans laquelle cette méthode s'applique (exemple : « toujours » ou « si le terminal la supporte ») et l'action en cas d'échec (passer à la prochaine méthode ou déclarer l'authentification du porteur comme échouée). Lors d'une transaction, le terminal lit le CVM List et applique les méthodes en fonction de ses capacités (présence ou non d'un clavier sécurisé pour la saisie du PIN, d'une imprimante pour imprimer la facturette et recueillir la signature du porteur, ...).

4.2.0.2.3 Authentification de l'émetteur

Lorsqu'une carte prend la décision d'autoriser la transaction *online* elle génère un cryptogramme appelé *AuthORIZATION Request Cryptogram* (ARQC). Ce cryptogramme est vérifié par l'émetteur ce qui lui permet d'authentifier la carte. Lors du processus d'autorisation, l'émetteur génère un cryptogramme appelé *Authorization Response Cryptogram* (ARPC) encapsulé dans l'élément de données *Issuer Authentication Data* (IAD). Les IAD sont ensuite envoyées à la carte qui en vérifiant le cryptogramme va alors authentifier l'émetteur.

Le chapitre suivant présente l'état de l'art des méthodes de tests de pénétration, des attaques sur les cartes à puce et sur les transactions de paiement.

4.3 Etat de l'art

4.3.1 Les méthodes de test de pénétration

Le test de pénétration est une méthode d'évaluation de la sécurité de solutions logicielles ou réseaux. On peut regrouper les méthodes de test de pénétration selon la classification suivante [127] :

- Le test en boîte blanche (*white-box*), aussi appelé test structurel, consiste à la compréhension du code source et de son architecture. Ce test s'avère efficace pour la découverte des erreurs de programmation et d'implémentation. Dans certains cas, cette analyse peut être automatisée grâce à des outils tels que *SourceScope* [128] et *IDA* [129]. Un des inconvénients de cette méthode est qu'elle peut révéler des vulnérabilités qui n'existent pas, appelées faux positifs.
- Le test en boîte noire (*black-box*), aussi appelé test fonctionnel, consiste à analyser un programme en exécution en variant ses données d'entrée et à observer l'effet sur les données de sortie afin d'y détecter d'éventuelles divergences par rapport à la spécification. Ce type de test ne nécessite pas le code source, seulement le programme en exécution, ce qui lui permet de réaliser une analyse à distance. Le test en boîte noire a également la particularité de nécessiter moins d'expertise que le test en boîte blanche.
- Le test en boîte grise (*gray-box*), aussi appelé test translucide, est une combinaison du test en boîte blanche et du test en boîte noire. Il requiert en général l'utilisation de plusieurs outils en même temps. Par l'exemple, un programme est exécuté en mode pas-à-pas dans un debugger tout en faisant varier ses données d'entrée afin de détecter un éventuel comportement suspicieux.

En conclusion, ces trois méthodes permettent de découvrir des vulnérabilités. Le test en boîte blanche permet d'identifier des bugs mais ne permet pas de mesurer les

risques de faille. Le test en boîte noire permet d'identifier ces failles. Le test en boîte grise combine ces deux méthodes et s'avère extrêmement puissant.

4.3.2 Attaques sur les cartes à puce et les transactions électroniques de paiement

4.3.2.1 Attaques classiques sur les cartes à puce

Depuis sa création à la fin des années 1970, la carte à puce a connu une grande évolution. Simple carte à mémoire à ses débuts, puis carte à microprocesseur et aujourd'hui assistée d'un coprocesseur cryptographique et d'un générateur de nombres aléatoires ; la carte s'est complexifiée tant au niveau matériel que logiciel. Aujourd'hui, une carte à puce embarque des plateformes logicielles complètes capables d'accueillir plusieurs applications écrites dans des langages publics et accessibles. Mais cette complexification s'est également traduite par un besoin accru de sécurité. La sécurité de la carte à puce est assurée par quatre composants [53] :

- le corps de la carte,
- les composants matériels de la puce,
- le système d'exploitation,
- l'application.

Chacun de ces composants est sensible à certaines attaques expérimentées ces deux dernières décennies. Les premières attaques avaient principalement trait au corps de la carte et aux composants de la puce. Parmi ces attaques, que nous appellerons *attaques classiques*, on référence trois principaux types d'attaques [130] :

- les attaques *invasives*, dans lesquelles le microprocesseur est retiré de la puce et directement attaqué ;
- les attaques *semi-invasives*, où la surface de la puce est exposée et soumise à des contraintes ;
- les attaques *non-invasives*, dans lesquelles on cherche à obtenir des informations sans apporter de modifications à la carte (par exemple, en observant les signaux électriques et émanations électro-magnétiques).

Puis, d'autres attaques ont vu le jour suite notamment à l'apparition des plateformes multi-applicatives ouvertes. Ces plateformes permettent de développer aisément des applications et de les installer sur une carte. Un attaquant peut alors charger une application malicieuse qui produira des attaques dites *internes*. Parmi ces attaques, on compte :

- l'identification de service,

- la collecte d'informations,
- les attaques contre les mécanismes de la plateforme.

De nouvelles classes d'attaques apparaissent ensuite. Ces attaques tirent à la fois partie des attaques classiques et internes, *i.e.* elles consistent en l'observation des signaux au moment de l'injection de fautes logicielles. Toutes ces attaques ont été traitées en profondeur dans un rapport de recherche rédigé dans le cadre de cette thèse [131].

4.3.2.2 Attaque par relais

Pour expliquer l'attaque par relais, prenons le célèbre exemple du Grand Maître d'échec comme discuté dans [132]. Dans ce scénario, une personne qui ne connaît pas les règles du jeu d'échec pourrait jouer contre deux Grands Maîtres par courrier interposé. Le joueur aurait simplement à transférer le coup joué d'un Grand Maître à l'autre. Il arriverait ainsi à faire jouer les deux Grands Maîtres l'un contre l'autre. Chaque Grand Maître pense jouer contre notre joueur inexpérimenté alors qu'en fait ils jouent l'un contre l'autre Grand Maître. L'application de ce scénario aux protocoles de sécurité a été présenté pour la première fois dans [133]. Dans la littérature, on trouve cette attaque sous le nom de « attaque du trou de ver » (*wormhole attack*) ou d'« attaque par relais » (*relay attack*) [134]. De nombreuses attaques par relais sur des cartes sans contact ou des mobiles NFC ont été implémentées. On prendra l'exemple de l'attaque menée par Francis *et al.* sur un mobile NFC [135]. Elle est illustrée à la figure 4.4.

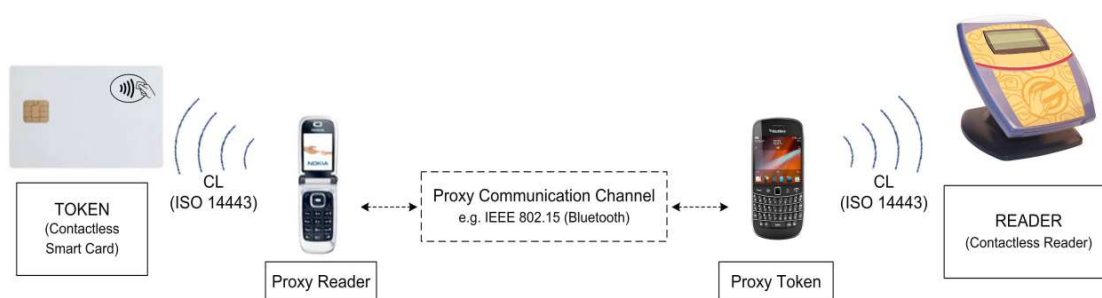


FIGURE 4.4 – Attaque par relais sur un mobile NFC

4.3.2.3 Attaque de Cambridge

L'attaque dite « de Cambridge » a été réalisée dans le *Computer Laboratory* de l'*Université de Cambridge* et exposée dans [136]. Dans cet article, Murdoch *et*

al. démontrent la faisabilité d'une attaque sur une carte à puce EMV. Il s'agit d'une attaque de type *man-in-the-middle* sur la vérification du code PIN (Personal Identification Number). Elle consiste à intercepter la commande VERIFY envoyée par le Terminal de Paiement Electronique (TPE) à la carte à puce afin de vérifier le code PIN saisi par le porteur de la carte. Elle s'appuie sur le fait qu'une carte à puce EMV peut supporter d'autres méthodes de vérification du porteur que le code PIN "offline" - *i.e.* vérifié par la carte à puce -, telles que la vérification de la signature du porteur et la vérification du code PIN "online" par la banque émettrice. Donc, si l'on réussit à intercepter la commande et envoyer une réponse positive au TPE à la place de la carte, à la fois le TPE et la carte "pensent" que la vérification du porteur a été un succès et que la transaction peut suivre son cours. Murdoch *et al.* ont mis en œuvre cette attaque de manière matérielle. Elle est composée d'une fausse carte à puce connectée à une carte FPGA. La carte FPGA est connectée à un ordinateur portable, qui est lui-même connecté par un lecteur de carte à puce à une carte à puce EMV volée (*cf.* Figure 4.5).

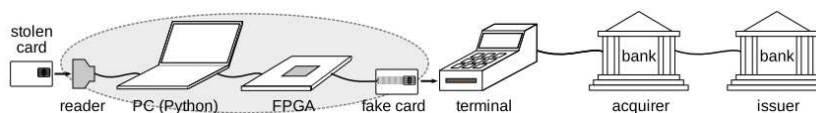


FIGURE 4.5 – Matériel utilisé pour mener l'attaque (source : [136] courtoisie de Murdoch *et al.*)

Un script python s'exécutant sur l'ordinateur portable relaie toutes les commandes envoyées par le TPE à la carte. Il attend une commande VERIFY et, au lieu de la relayer à la carte, renvoie le code 0x9000 au terminal. À ce stade, le point de vente estime que le code PIN a été vérifié avec succès par la carte et poursuit la transaction de paiement.

4.3.2.4 Attaque sur l'authentification EMV SDA

La faiblesse du mécanisme d'authentification EMV SDA (*Static Data Authentication*) réside dans le fait que les données d'authentification *offline* de la carte sont personnalisées et sont stockées de manière statique dans la carte. En analysant ce schéma, on se rend compte que pour n transactions, transiteront n fois les mêmes données entre la carte et le terminal puisqu'elles sont statiques. Ces données peuvent alors être utilisées pour cloner la carte même sans la connaissance de la clé privée de l'émetteur. Pour lutter contre cette faiblesse, ce mécanisme a été « banni » au profit de la méthode DDA (*Dynamic Data Authentication* (*cf.* section 4.2.0.2)).

4.3.2.5 Attaques induites par l'altération d'un terminal de paiement

Dans [137], Drimer *et al.* ont réussi à contourner les mécanismes de sécurité d'un terminal de paiement EMV. Cela leur a permis d'obtenir une copie du code PIN saisi par le porteur sur le clavier sécurisé. De plus, si la bande magnétique est utilisée pour mener à bien une transaction, alors elle est capturée, clonée et utilisée dans des pays n'acceptant pas le paiement par carte à puce.

4.3.2.6 Discussion

Dans le domaine de la sécurité, la reproductibilité des attaques de l'état de l'art est un facteur clé pour la compréhension des mécanismes d'attaques et la mise au point de nouveaux. Dans le domaine qui nous intéresse, les attaques sont souvent réalisées de manière matérielle ce qui engendre des coûts et nécessite des connaissances particulières. Par exemple, l'attaque implémentée par les chercheurs de Cambridge nécessite un FPGA et la connaissance de sa programmation. Ainsi, l'implémentation d'une plateforme d'analyse logicielle permet de se soustraire à ses contraintes. Les sections suivantes présentent la plateforme développée et sa mise en œuvre pour implémenter des tests lors des développements d'applications, et des attaques lorsqu'elles sont déployées et utilisées.

4.3.3 Plateformes d'analyse de transactions électroniques

Cette section présente un état de l'art des logiciels et bibliothèques permettant l'analyse de transactions.

4.3.3.1 Les logiciels

Gem_PCSC [138] de Gemalto permet d'agir au niveau des paramètres de connexion PC/SC ainsi que sur le protocole de transmission utilisé avec la carte (T=0 ou T=1). Il propose la sauvegarde des commandes exécutées sous forme de script afin de les rejouer ultérieurement. L'interface est intuitive et permet de voir toutes les informations nécessaires et de suivre l'état de la communication.

CardPeek [139] permet d'accéder et d'interpréter les informations personnelles stockées sur une carte à puce telles que l'historique des transactions de paiement. L'outil est actuellement capable de lire les cartes bancaires EMV, les cartes de transport Navigo, le porte-monnaie électronique Monéo, la carte de sécurité sociale Vitale 2, les passeports électroniques/biométriques conforme au standard ICAO (avec une sécurité de type BAC), les cartes SIM GSM ainsi que les cartes sans-contact Mifare. Enfin, détail intéressant, il propose un mécanisme d'envoi de commandes

manuelles sous forme de shell (en langage LUA) afin d'explorer des cartes non prévues nativement.

SmartCard ToolSetPro [140], de la société SCardSoft, est une solution très complète. En effet, la suite logicielle présente dans une interface graphique soignée des informations complètes sur le lecteur, la carte et le protocole utilisés. Elle permet l'envoi de commandes APDU une par une ou par lot et une analyse détaillée des APDUs de réponse.

4.3.3.2 Les librairies

SDK PC/SC de SpringCard [141] est un kit de développement permettant de s'interfacer aisément avec un lecteur respectant le standard PC/SC. Il intègre des librairies propriétaires à intégrer dans son propre projet de développement, quelques extraits de code source, des programmes d'exemple et une documentation complète. Les programmes d'exemple permettent de lire et écrire un tag NFC, d'explorer une carte de transport répondant à la norme *Calypso*.

SmartCard Framework [142], développé par Olivier Rouit, propose une librairie à intégrer dans un projet de développement pour faciliter le développement d'applications pour carte à puce. L'API expose des objets métier tels que `APDUCommand` et `APDUResponse` et des fonctions PC/SC élémentaires telles que `Connect`, `Disconnect`, `Transmit...` Cependant, son architecture n'est pas modulaire ce qui ne lui permet pas d'étendre ses fonctionnalités de base. De plus, il ne gère pas l'ensemble des protocoles de transmission prévus par la norme. Le code source est disponible sous licence CPOL [143] et un tutoriel est proposé.

PCSC-sharp [144], développée par Daniel Müller, est une autre librairie facilitant l'interface entre la carte et le lecteur. L'auteur a apporté un grand soin à mettre en place une architecture modulaire et évolutive, tout en proposant des objets permettant d'implémenter la norme sans restriction.

4.3.4 Discussion

Cet état de l'art souligne un certain manque de librairies disponibles pour s'interfacer avec un lecteur PC/SC et implémenter les protocoles ISO 7816 et EMV. Certaines des attaques sur les cartes à puce et les transactions électroniques de cet état de l'art ont la particularité d'être assez facilement reproductibles en « laboratoire » (exemple : l'attaque de Cambridge). La reproduction des attaques de l'état de l'art est une étape importante dans la recherche en sécurité, particulièrement dans des domaines aussi spécifiques que la carte à puce. Pour cette raison, nous avons

choisi d'implémenter une suite logicielle modulaire permettant de s'interfacer avec des lecteurs, d'implémenter le protocole ISO 7816 dans son intégralité, ainsi que le protocole EMV. Son architecture modulaire permettra d'implémenter tout type de protocole basé sur ISO 7816 (exemple : ICAO, Monéo, Calypso ...). Cette suite logicielle nous permettra donc de reproduire les attaques de l'état de l'art et d'en implémenter de nouvelles.

4.4 Le framework « WinSCard Tools »

Cette section détaille le framework WinSCard Tools (WSCT) développé à l'EN-SICAEN par Sylvain Vernois (mon co-encadrant de thèse). Tout au long de la thèse, je me suis appuyé sur cet outil et ai contribué à son amélioration et à son évolution.

4.4.1 Introduction

Lors du développement d'une application exploitant une carte à puce, le premier élément requis est une interface de programmation (*Application Programming Interface*, API) permettant d'accéder aux ressources du lecteur et à la puce elle-même.

Lors de nos recherches d'informations sur le développement pour cartes à puce, de nombreuses ressources ont attiré notre attention. Nombre d'entre elles impliquent le développement de logiciels embarqués dans la puce, la plupart utilisant JavaCard² [145]. Les sujets couvrent la modélisation d'applications embarquées jusqu'aux problèmes de conformité avec les spécifications [146]. La question de la sécurité et la robustesse des cartes est un autre domaine d'étude. Cependant, nos besoins ne sont ni le développement d'une application embarquée ni la validation de la plateforme, mais de disposer d'outils facilitant l'accès aux ressources des cartes à puce à partir d'un ordinateur, afin de développer des applications permettant l'étude de multiples aspects d'une application embarquée sur une carte à puce.

Pour ce besoin, nous aurions pu utiliser une API propriétaire, fournie par le fabricant du lecteur de carte, avec pour défaut majeur le fait de ne pas être réutilisable si le lecteur de carte venait à changer. Une autre solution consiste à utiliser une API publique et reconnue telle que PC/SC [147], l'abstraction du lecteur étant fournie.

2. JavaCard définit un environnement d'exécution standard pour carte à puce utilisant une machine virtuelle java embarquée dans la puce, permettant à une applet de fonctionner sur des cartes à puce techniquement différentes.

4.4.1.1 La spécification PC/SC

Cette spécification a été écrite pour faciliter l'utilisation de la technologie des cartes à circuits intégrés dans un environnement PC (*Personal Computer*). Les principaux contributeurs sont Apple, Axalto, Gemplus, Hewlett-Packard, IBM, Infineon, Ingenico, Microsoft, Philips, Siemens, Sun Microsystems, Toshiba and VeriFone, réunis au sein du groupe de travail PC/SC [148].

4.4.1.2 Les implémentations PC/SC

4.4.1.2.1 Systèmes d'exploitation Windows

La première implémentation de la spécification PC/SC a été réalisée par Microsoft pour le système d'exploitation Windows. Fournie comme un logiciel tierce partie pour les systèmes Windows NT/9x, cette spécification est systématiquement implémentée dans toutes les versions depuis Windows 2000. L'implémentation consiste, en particulier, en une librairie dynamique, appelée `winscard.dll`, définissant un jeu de fonctions élémentaires. Toutes les informations à destination des développeurs sont publiées sur le site MSDN [149].

4.4.1.2.2 Systèmes d'exploitation Unix/Linux

MUSCLE [150], signifiant *Movement for the Use of Smart Cards in a Linux Environment*, a deux objectifs principaux :

- Fournir un *middleware* pour accéder à une carte à puce en utilisant l'API PC/SC (projet 'pcslite')
- Fournir une framework permettant la réalisation de solutions indépendantes de la plateforme et de la carte (projet 'musclecard')

Le projet 'pcslite' fournit une API très similaire de celle délivrée par Microsoft.

4.4.1.3 Le besoin d'une API simple

Les implémentations existantes permettent l'utilisation d'un grand nombre de lecteurs de cartes provenant de la majeure partie des fabricants, à condition qu'un pilote conforme soit fourni. Ces APIs, bien que très utiles car fournies sous formes de fonctions C, sont inadaptées à l'usage des "langages modernes" orientés objet.

4.4.1.3.1 Les bases d'une nouvelle API

Parce que le langage C# est moderne, robuste et portable sur plusieurs systèmes (*plateforme .net* pour les systèmes d'exploitation Windows et *projet mono* pour les

systèmes Linux), il est utilisé dans plusieurs projets de recherche de l'équipe. Nous l'avons également choisi pour développer notre API orientée objet basée sur PC/SC. Les objectifs suivants ont été pris comme fil conducteur pour le développement de l'API :

1. Publier une API orientée objet simple et facile d'utilisation ;
2. Publier une API étendue offrant plus de contrôle sur la communication entre la carte et l'application ;
3. Permettre l'indépendance entre l'application cliente et la carte.

Une API respectant ces contraintes a été développée dès 2006. Aujourd'hui, elle est accompagnée d'une interface graphique qui s'avère être d'une grande aide pour les tests et développements futurs.

Dans les sections suivantes, nous détaillons l'architecture de l'API *WinSCard Tools* et du logiciel l'accompagnant. Ensuite, nous présentons des outils complémentaires développés grâce à l'API et ayant attrait plus spécifiquement aux cartes de paiement. Enfin, nous montrons comment ces outils peuvent être utilisés pour la recherche sur les cartes à puce en rejouant des attaques connues sur une transaction EMV.

4.4.2 Présentation de « WinSCard Tools »

4.4.2.1 Une architecture ouverte

Le logiciel développé est divisé en deux parties, chacune réalisant une fonction spécifique. La première est une interface de programmation orientée objet encapsulant la bibliothèque dynamique native PC/SC Windows `winscard.dll` ou `libpcsc.so` sous linux. La seconde est une interface utilisateur de base, visant à faciliter l'utilisation du lecteur de carte à puce et à servir de support au développement d'applications de test en tant que plug-ins.

4.4.2.2 L'API pour cartes à puce

Les trois objectifs précédemment mentionnés conduisent naturellement à la mise en œuvre de trois modules respectifs. La Figure 4.6 décrit les interactions entre ces modules, basés sur la machine virtuelle .net et l'implémentation native PC/SC. Le module `Wrapper` permet à la plateforme .net d'accéder aux fonctions natives grâce à des méthodes d'appel de code "non managé". Il est chargé de résoudre les problèmes de communication qui peuvent survenir entre la bibliothèque écrite en C et le langage C#. Il est principalement composé de deux classes : `Primitives` est une

classe statique réalisant l'interopérabilité avec la bibliothèque native et `ErrorCode` est une énumération des codes de retour de la couche PC/SC. Le module `Core` est plus intéressant, car il publie plusieurs interfaces et classes qui peuvent être utilisées de manière flexible, comme illustré figure 4.7. Un objet `CardContext` permet la prise et la libération des ressources PC/SC de l'ordinateur, et doit être unique dans une application. Une fois le contexte établi, les lecteurs de carte à puce installés sur le système sont disponibles et prêts à être utilisés via un objet `CardChannel`, qui permet la communication directe entre l'application et le lecteur de carte ou la carte à puce. Les interfaces de ces classes sont isolées car, comme il sera montré plus tard, plusieurs implémentations peuvent coexister et hériter de l'original.

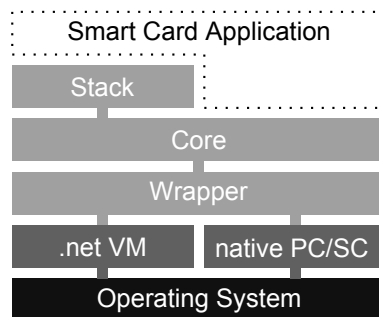


FIGURE 4.6 – Architecture générale de l'API

L'objectif d'avoir un contrôle renforcé sur la communication est réalisé grâce au design pattern "observateur" [151], décrivant comment un objet qualifié d'"observable" peut publier des informations internes à tout objet qui s'est préalablement enregistré auprès de lui. Ce mécanisme est la source de la flexibilité de notre API, et y est largement utilisé. En outre, le langage C# offre un moyen très simple de mise en œuvre de ce modèle par l'utilisation du mot-clé `delegate`. La déclaration explicite d'une interface pour chaque information à publier étant évitée, le code exploitant le mécanisme d'observation reste succinct et clair.

4.4.2.2.1 La pile de communication

En 2004, une première implémentation en Java de la spécification ISO FDIS/IEC 24727-2 [152] a été développée à l'ENSICAEN, connue sous le nom de `Cardstack`. La nouvelle API développée a bénéficié de cette expérience et acquis une pile conforme à la norme. De cette façon, le troisième objectif initial a été complété. Concrètement, ce qui a été réalisé est un ensemble de classes (comme le montre la Figure 4.8) permettant d'ajouter des niveaux d'abstraction entre l'application et la carte à puce. Même si une carte à puce est conforme à la norme ISO/IEC 7816 [153], décrivant les exigences

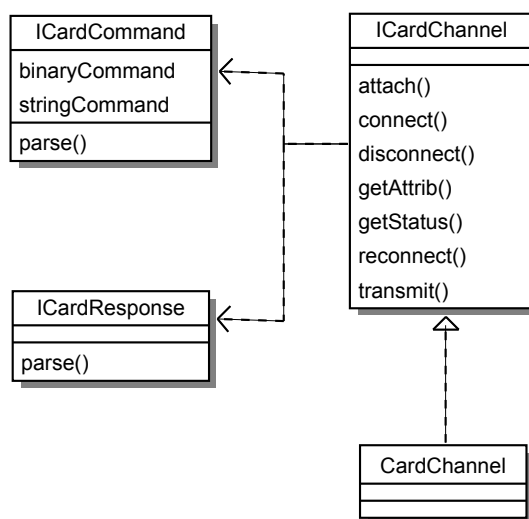


FIGURE 4.7 – Diagrammes de classe partiels du module "Core"

en matière de cartes à circuits intégrés, elle n'implémente pas nécessairement toutes les parties de la norme. C'est pourquoi une application est rarement capable de supporter toutes les cartes à puce, mais seulement un sous-ensemble, ou doit alors être adaptée en cas de changement de carte à puce. L'abstraction ajoutée par la pile de communication permet de réduire ces difficultés d'interopérabilité, en permettant à l'application d'être indépendante de la carte à puce utilisée : une couche d'adaptation intermédiaire peut être mise en place entre l'application et la puce pour modifier à la volée les commandes envoyées à la carte et les réponses qui sont reçues.

Parce que `CardChannel` et `CardChannelStack` implémentent la même interface, l'un peut être utilisé en remplacement de l'autre sans aucune difficulté, permettant l'ajout d'une pile très facilement à une application sans aucune modification du code, sauf lors de l'instanciation de la classe. Chaque couche de la pile doit implémenter l'interface `ICardChannelLayer` afin qu'elle puisse être insérée dans la pile. Une couche spécifique `CardChannelLayer` hérite de l'objet `CardChannel` pour permettre son insertion dans une pile.

4.4.2.3 Une interface utilisateur

Un outil d'accompagnement, servant de preuve de concept, a été réalisé simultanément à la pile de protocole. Il a évolué vers une interface graphique (GUI) exploitant toutes les capacités de l'API et permettant une extensibilité grâce à un système de plugin, comme illustré à la figure 4.10.

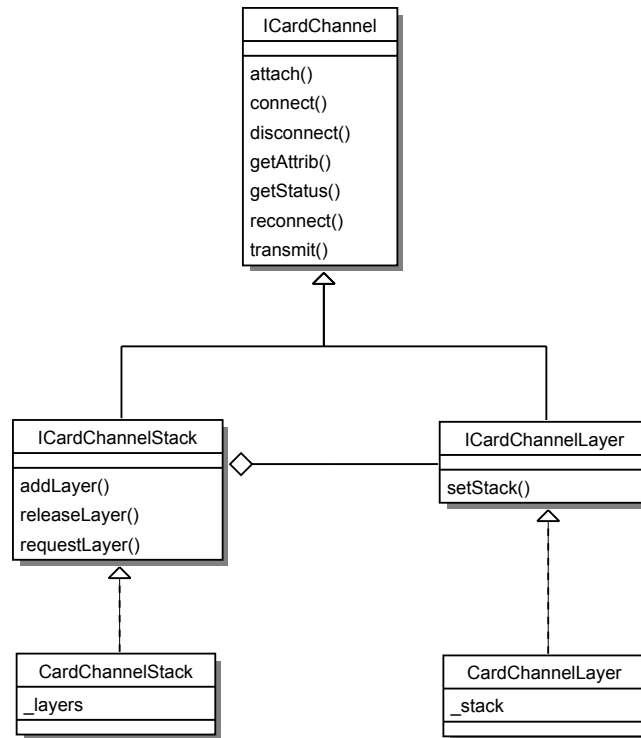
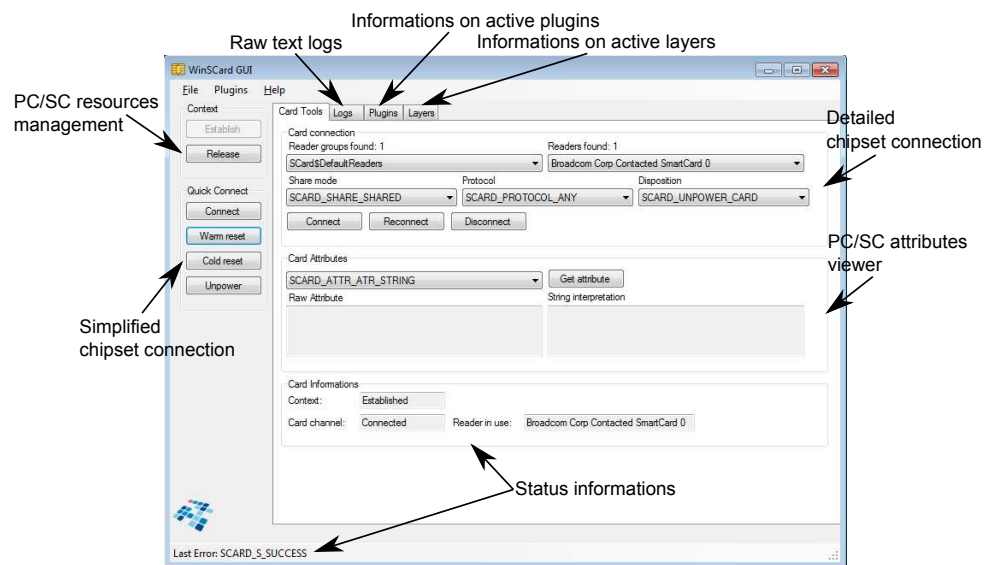


FIGURE 4.8 – Diagramme de classe partiels du module "Stack"

FIGURE 4.9 – Fenêtre principale de l'interface graphique de *WinSCard Tools*

4.4.2.3.1 L'interface graphique principale

L'interface graphique principale (cf. figure 4.9) est responsable de la gestion des ressources : accès aux pilotes PC/SC sous-jacents et gestion du canal établi entre

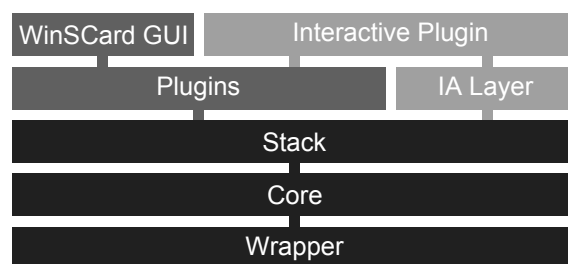


FIGURE 4.10 – Architecture de l’interface graphique

l’utilisateur et la carte elle-même.

4.4.2.3.2 Le système de plugins

Un système de plugins est utilisé pour étendre les fonctionnalités de l’interface graphique, afin d’implémenter facilement des petites applications indépendantes (les plugins) répondant à un besoin bien précis. Tous les plugins et l’interface graphique principale partagent un contexte et un canal uniques offerts par l’API. En utilisant activement les mécanismes de communication par évènement, plusieurs plugins peuvent, de manière transparente, accéder aux données échangées entre l’application cliente et la carte de manière transparente, offrant ainsi la possibilité d’analyser et de modifier les données échangées ”à la volée”.

4.4.2.3.3 Enregistrement et rejeu d’échanges APDU

En utilisant l’interface graphique pour des projets éducatifs et de recherche, nous avons rencontré à un besoin récurrent : pouvoir lire et rejouer un ensemble de réponses provenant d’une carte pour permettre des analyses multiples. Le même jeu de commandes peut être envoyé à plusieurs reprises à la même carte, mais dans notre cas, cela ne suffit pas car une transaction met en jeu des nombres aléatoires générés et des compteurs incrémentés par la carte. Le même scénario ne peut donc être reproduit. C’est pourquoi une nouvelle couche, désignée comme “Interactive layer” sur la figure 4.10 a été ajoutée. En utilisant le mécanisme de pile, cette couche est introduite entre l’application et la carte à puce et propose trois modes (*cf.* figure 4.11) :

- Le mode *record* permet de mémoriser et d’enregistrer les données échangées.
- Le mode *replay* permet de rejouer les données précédemment mémorisées. Dans ce mode, la carte n’est plus physiquement accédée, mais la couche Interactive renvoie les données directement à l’application, qui n’a pas connaissance de la ”fausse” carte.
- Le mode *transparent* agit en ”pass through”, sans action sur la communication.

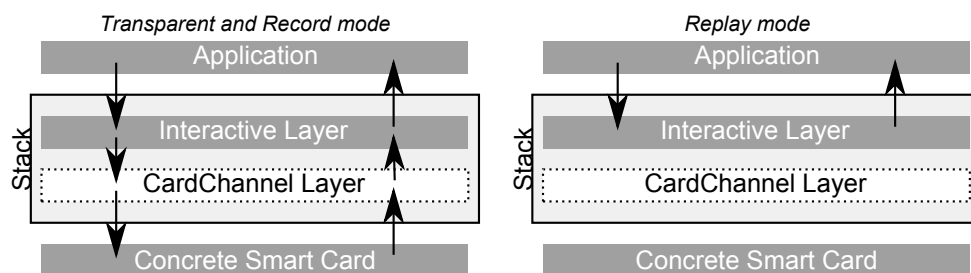


FIGURE 4.11 – Illustration des modes de fonctionnement de la couche Interactive

Dans cette section, nous avons présenté l'API et les outils graphiques créés pour simplifier l'accès aux ressources de la carte à puce. L'architecture ouverte, basée sur des modèles de conception simple, permet non seulement la communication entre une application (ou un plugin) et la carte, mais fournit également des moyens pratiques pour analyser et modifier les données.

4.4.3 Application à l'analyse des transactions

4.4.3.1 Le plugin 7816 : une première implémentation concrète

Le besoin initial pour *WinSCard Tools* est venu de notre principal domaine de recherche basé sur les paiements électroniques, où les cartes à puce sont souvent utilisées. L'utilisation de l'APDU³ normalisée ISO/IEC 7816 [153] nous a conduit à ajouter un plugin spécifique dédié à ces cartes. Parmi les fonctionnalités, la plus importante est la capacité d'observer la communication et de décoder toutes les paires de commande/réponse comme APDUs standards, et ce de manière non intrusive. L'APDU est ensuite stockée, et l'historique est conservé pour une utilisation ultérieure. Une fois qu'une transaction est complétée, toutes les données peuvent être sauvegardées sur le disque, ou rejouées unitairement à volonté. Le plugin est également en mesure d'envoyer des commandes enregistrées ou nouvelles depuis sa propre interface graphique (*cf.* figure 4.12). Ce plugin est d'une grande aide lorsque l'on s'intéresse aux données brutes échangées lors d'une transaction.

4.4.3.2 Une librairie EMV à des fins de recherche

EMV⁴[54] est une spécification fréquemment utilisée pour les transactions de paiement par cartes à puce (*cf.* section 4.2), particulièrement parce qu'elle est imposée

3. Application Protocol Data Unit : représente la commande et la réponse échangés à l'interface

4. Europay MasterCard Visa

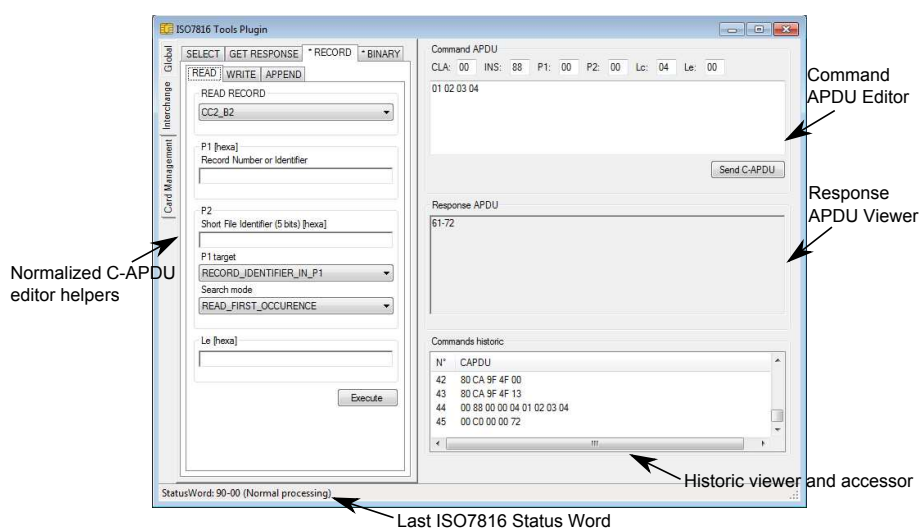


FIGURE 4.12 – Interface graphique du plugin ISO 7816

par les principaux réseaux de paiement dans le monde entier : Mastercard, Visa, JCB et American Express. C’est pourquoi nous avons mis en place une bibliothèque EMV, basée sur l’API développée, avec un plugin dédié pour gérer la transaction. Cette bibliothèque a été réalisée avec les mêmes objectifs que l’API : en particulier, elle expose plusieurs *événements* pour surveiller les fonctionnements internes.

Le plugin associé permet de piloter une transaction EMV, étape par étape, et d’analyser avec précision toutes les APDU conformes aux spécifications EMV, comme le contrôle des cryptogrammes générés par la carte. Aussi, certains composants ont été adaptés pour permettre à certaines cartes sans contact basées sur EMV d’être utilisées. Un exemple d’utilisation à des fins de recherche de la bibliothèque EMV est détaillé dans la section suivante.

4.4.3.3 Rejeu de l’attaque de « Cambridge »

4.4.3.3.1 Implémentation logicielle de l’attaque

A des fins de recherche et d’enseignement, nous avons reproduit cette attaque de manière logicielle grâce à *WinSCard Tools*. En effet l’architecture, de *WinSCard Tools* permet de définir très facilement une couche ”man-in-the-middle” (MITM) pour implémenter cette attaque.

Dispositif expérimental La couche MITM est ajoutée à la pile de communication et relaie toutes les commandes envoyées par *WinSCard Tools* à la carte, à l’exception de VERIFY (*cf.* figure 4.13). Au lieu de cela, il renvoie 0x9000 à

WinSCard Tools. Ensuite, la transaction continue normalement et on peut observer que la carte valide la transaction.

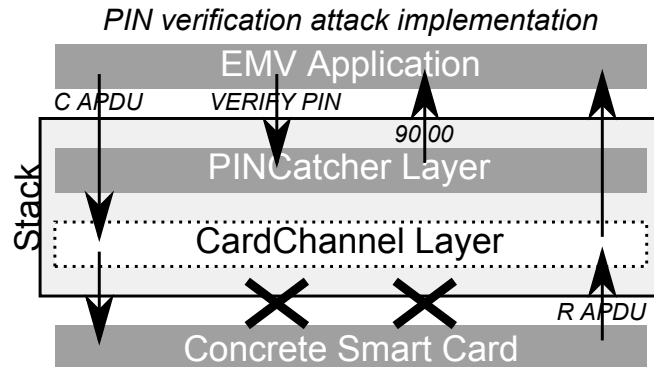


FIGURE 4.13 – Implémentation logicielle de l'attaque sur le code PIN

Grâce à *WinSCard Tools*, il est possible d'observer ce qui se passe du côté du TPE lors d'une transaction. En effet, dans une transaction EMV, il est possible d'espionner le lien entre la carte et le TPE, mais il n'est pas possible de connaître la façon dont le POS traite les données, par exemple, la façon dont le TPE vérifie la validité des certificats de la carte à puce et les données qu'il en extrait. Comme *WinSCard Tools* émule complètement un TPE et embarque la librairie EMV vue précédemment, ces traitements sont totalement accessibles à l'utilisateur. Une interface utilisateur dédiée présente les différents certificats, le résultat du processus de vérification de signature et les données contenues dans les certificats.

Une autre fonction de *WinSCard Tools* est particulièrement utile pour rejouer les attaques : la couche Interactive. Cette couche permet d'enregistrer et rejouer les transactions. Elle enregistre les données envoyées par la carte à la réception des commandes et peut les rejouer au nom de la carte en mode de lecture. Dans la mise en œuvre de ce genre d'attaque, cette fonctionnalité est intéressante car elle permet de conserver, par exemple, le compteur du nombre de transactions inchangé. En effet, pour chaque transaction initiée, un compteur est incrémenté. Ce compteur est utilisé, avec d'autres éléments, par la carte pour décider si le terminal doit interroger la banque pour autoriser la transaction. Il est alors intéressant de maintenir inchangé les données d'une carte à puce EMV véritable lorsque l'on effectue une certaine quantité de tests et d'attaques.

Mode opératoire Une fois l'environnement expérimental établi, une carte à puce EMV est insérée dans le lecteur et le plugin "EMV Explorer" est lancé. EMV Explorer (cf. Figure 4.14) simule un point de vente et permet de traiter une

transaction étape par étape. Il affiche toutes les commandes APDU échangées entre la carte et le terminal et les interprète grâce à la bibliothèque EMV.

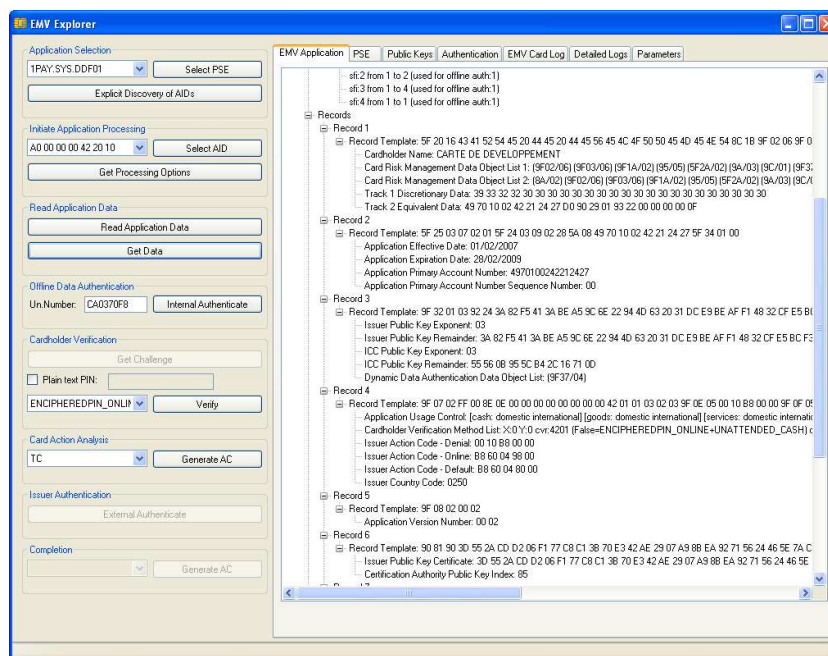


FIGURE 4.14 – Le plugin ”EMV Explorer”

A l'étape de la vérification du porteur, un code PIN est saisi et on peut observer que le POS reçoit le mot 0x9000 (traitement normal) puis procède à la commande suivante (Generate AC).

Résultats et discussion L'attaque a été effectuée avec succès sur toutes les cartes à puce EMV en notre possession (de vraies cartes et des cartes de test). Aucune d'elles n'a été capable de détecter l'attaque, car elles permettent d'autres CVM que la vérification du code PIN. Ceci est également dû au fait que les émetteurs n'ont pas implémenté de contre-mesure dans les versions des applications que nous avons sur nos cartes à puce. Mais l'implémentation d'une contre-mesure est prévue dans les nouvelles spécifications d'application - de Visa et MasterCard, par exemple. Elle consiste souvent à passer l'élément de données *CVM Result* à la carte lors de l'envoi de la commande GENERATE AC. Pour protéger la transmission du *CVM Results* d'une autre attaque man-in-the-middle, nous recommandons d'utiliser CDA (Combined DDA/Application Cryptogram Generation) pour signer de manière dynamique les données envoyées à la carte dont le CVM Result.

Nous avons vu dans cette section comment nous avons été en mesure de reproduire

l'attaque dite "de Cambridge" de manière logicielle. Bien sûr, ceci ne peut pas être mis en œuvre dans la vie réelle avec de véritables points de vente tels que les chercheurs de Cambridge l'ont fait, mais nous avons démontré que *WinSCard Tools* est un outil parfaitement adapté à un travail de recherche sur les cartes à puce en général et sur les transactions EMV en particulier.

4.4.3.4 Interception et modification d'une transaction de paiement

L'attaque de Cambridge a démontré qu'une attaque de type *man-in-the-middle* sur une transaction de paiement EMV – qui était connue depuis quelques temps déjà – n'était pas que le fruit de recherches théoriques mais avait maintenant une implémentation réelle sur le terrain. En effet, l'équipe de Ross Anderson, après avoir publié ses résultats, a fait l'objet de nombreux articles de presse [154, 155] et d'un reportage vidéo [156] dans lequel la démonstration de son attaque chez un marchand est réalisée. Grâce à *WinSCard Tools*, nous faisons également la démonstration de la faisabilité de telles attaques mais avec un dispositif beaucoup plus réduit. En effet, le matériel constitué d'une carte FPGA et le logiciel en python pour l'analyse des APDUs sont regroupés en un seul outil logiciel. La preuve est également moins risquée en utilisant *Winscard Tools* que d'aller chez un véritable commerçant. L'attaque que nous avons choisie d'implémenter est une interception et une modification des données d'une transaction de paiement sans-contact.

Dans les sections suivantes, nous étudions le dispositif expérimental, discutons les résultats obtenus et concluons en proposant une amélioration significative à ce dispositif.

4.4.3.4.1 Dispositif expérimental

Le dispositif est constitué de l'outil *WinSCard Tools* auquel sont raccordés un lecteur de carte à puce et une sonde sans contact. Une vraie carte de paiement sans-contact est posée sur le lecteur tandis que la sonde est présentée à un terminal de paiement sans contact (*cf.* figure 4.15). Un plugin spécialement développé relaie les commandes du terminal à la carte et vice versa. Nous utilisons ce dispositif non plus pour faire du simple relais de commande mais pour les intercepter et les modifier.

4.4.3.4.2 Mode opératoire

Une fois le dispositif mis en place, on pose sur le lecteur une carte de paiement sans contact et la sonde sur le terminal de paiement. La carte implémente une méthode d'authentification des données statique appelée SDA (*Static Data Authentication*). Le terminal récupère de la carte le certificat émetteur (*i.e.* la clé publique de l'émetteur



FIGURE 4.15 – Dispositif d'attaque

signée par l'autorité de certification) et les données signées. Il vérifie alors le certificat, en extrait la clef publique et vérifie la signature des données. Cette vérification statique des données est sensible au rejeu et au clonage de la carte car ce sont toujours les mêmes données qui sont échangées entre la carte et le terminal.

Nous avons réussi à mettre en œuvre cette attaque grâce au plugin *Man in the middle*. Nous avons réalisé un certain nombre de transactions jusqu'à ce que le compteur du nombre de transactions consécutives réalisées *offline* arrive à son plafond. A présent, la carte refuse toute transaction. Grâce au plugin, nous interceptons les données renvoyées par la carte et les modifions afin de faire "croire" au terminal que la transaction est acceptée. Si cette attaque était réalisée chez un marchand, ce dernier délivrerait le bien acheté alors que la carte avait l'intention de refuser la transaction.

Pour obtenir l'autorisation de paiement le terminal demande à la carte de générer un cryptogramme accompagné d'un élément d'information indiquant la décision prise par la carte. Trois décisions sont possibles : refus de la transaction (*Application Authentication Cryptogram*, AAC), acceptation *offline* de la transaction (*Transaction Certificate*, TC) ou demande d'autorisation à la banque (*Authorisation Request Cryptogram*, ARQC). Le plugin intercepte les éléments retournés par la carte et modifie l'information sur le cryptogramme afin de modifier le type de cryptogramme retourné d'AAC à TC (figure 4.16).

4.4.3.4.3 Résultats expérimentaux

Avec ce dispositif, nous avons réussi à modifier les données d'une transaction afin que celle-ci soit acceptée par le terminal alors que la carte la déclinait. Pour cela, nous avons modifié l'octet codant le type de cryptogramme retourné (*A0* modifié

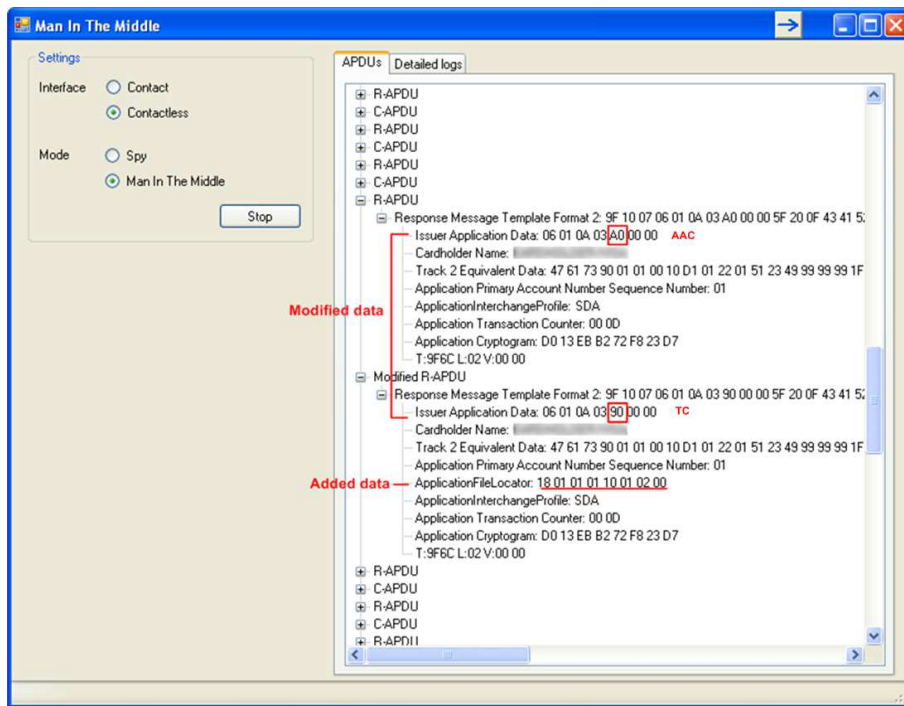


FIGURE 4.16 – Modification du type de cryptogramme retourné par la carte

en 90) et inséré une donnée toujours présente (avec l'application ciblée) lorsque que la transaction est acceptée *offline* : l'*Application File Locator*. Nous avons ainsi reproduit de manière logicielle, avec WinSCard Tools et une sonde, une attaque bien connue sur les transactions de paiement utilisant le mécanisme d'authentification des données EMV SDA.

4.4.3.4.4 Discussion

Cette attaque s'avère efficace sur les cartes de type SDA. Pour lutter contre le rejeu des transactions, depuis le 1er janvier 2010, toutes les cartes bancaires émises en Europe doivent implémenter une authentification dynamique appelée DDA (*Dynamic Data Authentication*). Contrairement à SDA, les données signées ne sont pas personnalisées dans la carte mais générées de manière dynamique à chaque transaction. La signature dynamique est générée à l'aide de la clef privée de la carte ce qui induit que celle-ci implémente l'algorithme RSA, par le biais d'un processeur cryptographique par exemple. Nous pourrions toujours modifier les données échangées mais la vérification de la signature dynamique des données permettrait de détecter l'attaque.

Dans le plugin *Man in the middle*, l'implémentation de l'attaque est codée "en dur" dans le code source et de manière spécifique au type de carte visée. Nous comptons

améliorer le plugin de manière significative en décrivant dans un fichier XML des heuristiques d'attaque sur les cartes à puce afin que le logiciel s'adapte automatiquement à l'application avec laquelle se réalise la transaction et qu'il implémente les attaques applicables en fonction du contexte.

Dans de futurs travaux, nous envisageons également de porter *WinSCard Tools* sur un équipement portatif de type assistant personnel numérique équipé de la plateforme Windows CE. En effet, comme Windows CE embarque la machine virtuelle .net, ceci nous permettrait de réduire de manière considérable l'encombrement du dispositif d'attaque. Le lecteur de carte sans contact pourrait également être remplacé par l'intégration de la technologie *NFC* (*Near Field Communication*) [14] car cette dernière permet à un équipement d'agir comme un lecteur de tags ou de cartes sans-contact RFID [35].

Dans cette approche, l'attaque a été implémentée à partir de la connaissance d'une vulnérabilité connue dans le protocole (échange de données statiques permettant le rejeu d'une transaction). Dans la section suivante, nous investiguons la possibilité de découvrir une nouvelle vulnérabilité à l'aide d'une technique connue dans l'état de l'art sous le nom de "fuzzing".

4.5 Un framework de fuzzing pour les services de paiement mobile

Dans cette section, nous présentons la conception d'un framework de fuzzing afin de tester les services de paiement. Nous présentons tout d'abord l'architecture du framework puis son utilisation dans l'amélioration de l'approche de Lancia [157]. Ce framework peut être utilisé aussi bien en boîte blanche qu'en boîte noire ou grise.

4.5.1 Découverte de vulnérabilités par fuzzing

Le « fuzzing » est une technique consistant à découvrir des vulnérabilités dans un programme ou dans un système. Le principe est d'injecter des données malformées ou aux limites de leurs valeurs. Les vulnérabilités peuvent être classées de différentes façons. Dans [158], Dowd *et al.* proposent une classification en trois classes :

vulnérabilités liées à la conception : introduites lors de la transcription du cahier des charges en spécifications fonctionnelles, elles-mêmes transcrites en spécifications techniques ;

vulnérabilités liées à l'implémentation : introduites au moment du développement du code du programme ou du serveur ;

vulnérabilités liées au fonctionnement : observées lors du déploiement et du fonctionnement du programme ou du serveur.

Les erreurs d'implémentation sont la plupart du temps détectées par des messages d'erreur ou d'avertissement du compilateur, des tests unitaires ou encore des tests d'intégration. Il se peut néanmoins que certaines erreurs ne soient pas détectées, soit par manque de ressources à allouer à la phase de test soit parce que la phase de test se concentre principalement sur la conformité aux exigences fonctionnelles. Par exemple, l'attaque sur les cartes Mifare Classic réalisée par Koning *et al.* et publiée dans [159], exploite une faiblesse du générateur pseudo-aléatoire impliqué dans le chiffrement de la communication entre la carte et le lecteur.

Dans le cadre des services mobiles sans contact, des erreurs d'implémentation peuvent avoir des conséquences désastreuses. Prenons l'exemple d'un service de paiement mobile de proximité présent dans le Secure Element sous la forme d'une applet. Cette application a été développée selon une spécification définie par un acteur tel que Visa, MasterCard, American Express ou selon une spécification dite propriétaire. Pour pallier à la difficulté de réaliser des tests exhaustifs, les techniques de fuzzing peuvent s'avérer d'une grande utilité. Dans les sections suivantes, nous décrivons les différentes méthodes de fuzzing et de génération de données d'entrée, le format des données d'entrée d'une applet, l'état de l'art des méthodes de fuzzing appliquées à la carte à puce et nous exposons l'approche proposée.

4.5.1.1 Les méthodes de fuzzing

On considère aujourd'hui que Barton Miller, de l'Université du Wisconsin, est le « père » du fuzzing. Il introduit le concept de « *fuzz program* » en 1988 dans un sujet de projet donné à des étudiants [160] et les premiers travaux furent publiés dans [161]. Il s'agissait alors d'attaquer des processus UNIX en leur envoyant des chaînes de caractères aléatoires afin de les faire « crasher ». Dans [162], Clarke expose les caractéristiques communes à la plupart des programmes de fuzzing :

- génération de données, création des données à envoyer en entrée selon une méthode pré-définie (*cf.* paragraphe suivant) ;
- transmission de données, envoi à la cible des données précédemment générées ;
- surveillance de la cible, observation de l'effet des données envoyées sur la cible ;
- automatisation, enchaînement programmé de la génération de données, de leur transmission et de la surveillance de la cible .

Les deux dernières peuvent être considérées comme optionnelles ou peuvent être implémentées dans un module externe au fuzzer. On distingue trois grandes méthodes

de génération de données [163] : aléatoire, par mutation de données et par analyse du protocole.

La génération aléatoire met en jeu un générateur qui produit un jeu de données de tests. Cette approche minimise les efforts et le temps requis mais s'avère être la moins efficace pour détecter des vulnérabilités car elles ne sont pas forcément connues à l'avance. La mutation de données combine deux techniques : la capture de données et la mutation sélective. A partir d'une donnée d'entrée valable, on effectue des mutations afin d'obtenir un jeu de données de test qui est très proche de la structure d'une donnée valide. La génération de données basée sur l'analyse du protocole s'appuie sur le principe que les données en entrée répondent très souvent à un protocole ou un format pré-défini. Un fuzzer de ce type est implémenté en définissant un modèle du protocole en question afin qu'il puisse créer des données d'entrée valides mais dont les données sont aléatoires.

Dans la section suivante, nous détaillons le dispositif expérimental, *i.e.* le format des données d'entrée de la partie d'un service mobile sans contact hébergée dans le Secure Element (applet).

4.5.1.2 Le fuzzing appliqué à la carte à puce

Dans [164], Guyot expose l'utilisation du fuzzing sur des applications pour cartes à puce. Il montre comme il est aisé de déterminer les commandes acceptées par l'application, *i.e.* l'octet de code instruction *INS* reconnue par l'application. Un code instruction reconnu provoquera une action et le retour de données ou d'un code de statut indiquant une erreur interne. Par exemple, l'applet peut retourner *6F00* si les données en entrée ne sont pas conformes et ont provoqué une erreur de traitement dans l'applet. En revanche, si le code instruction n'est pas reconnu, le code de statut standard *6D00* est retourné. Guyot utilise ensuite les résultats observés pour rendre l'applet (application de carte étudiant) « *fuzzing-proof* ». Ainsi, plutôt que d'utiliser le code instruction *INS* pour différencier les commandes, il utilise un *INS* fixe et différencie les commandes selon les paramètres de référence *P1* et *P2*. On observe une augmentation de la complexité pour un attaquant pour découvrir le fonctionnement de l'applet car l'utilisation des paramètres *P1* et *P2* pour différencier les commandes n'est pas « standard ».

Dans [165], Barreaud *et al.* exposent une méthode d'analyse de vulnérabilités sur carte à puce à serveur web intégré. Ils se proposent de fuzzer le protocole BIP (*Bearer Independent Protocol*) responsable de la communication au sein d'un téléphone mobile entre le processeur d'application et une carte à puce telle que la SIM. Ils utilisent le framework de fuzzing Peach qu'ils étendent avec la librairie Pyscard chargé de la

transmission de données vers la carte à puce. Ils modélisent le protocole BIP à l'aide d'un fichier XML et utilisent ce même fichier pour surveiller la cible en ajoutant deux balises : *<Expected>* et *<Response>*. Avec ce dispositif, ils découvrent que certaines des cartes testées n'implémentent pas le protocole correctement tel que défini par l'ETSI, que certaines possèdent des erreurs d'implémentation et enfin, ils découvrent qu'il est possible de réaliser un déni de service.

Dans [157], Lancia publie probablement la seule implémentation connue à ce jour d'une méthode de fuzzing visant à découvrir des vulnérabilités sur les applications EMV. Il utilise pour cela le framework de fuzzing par bloc Sulley (*i.e.* fuzzing par modélisation des données du protocole) [166]. Sulley est écrit dans le langage Python et est reconnu comme l'un des frameworks les plus complets et efficaces. Afin d'assurer la transmission de données, Lancia a intégré la librairie Triton, également écrite en Python et permettant de communiquer avec des cartes à puce par l'interface PC/SC. Pour la surveillance de la cible, Lancia a développé une implémentation de référence des spécifications des cartes EMV qu'il a choisi de cibler. Les mêmes commandes sont envoyées simultanément à la carte et au modèle de référence. Une anomalie est détectée lorsque le résultat renvoyé par la carte diffère de celui renvoyé par l'implémentation de référence. Toutes les commandes du protocole sont modélisées puis liées entre elles afin de modéliser le graphe d'état du protocole (cf. figure 4.17).

Pour tester une commande en particulier, toutes les commandes précédentes sont envoyées avec des valeurs par défaut. Pour une commande, le framework Sulley génère des données à partir du modèle de protocole défini. Ainsi, Lancia a pu mettre en évidence des écarts fonctionnels par rapport à la spécification et des failles sécuritaires sur des implémentations des spécifications Visa VIS et MasterCard M/Chip. Par exemple, il a constaté qu'une certaine combinaison de données engendrait la remise à zéro des compteurs *offline*.

L'approche de Lancia a montré son efficacité en remontant des écarts fonctionnels et sécuritaires sur des implémentations préalablement certifiées. Cette efficacité est due à une description très précise du modèle de données et à une grande minutie dans le développement de l'implémentation de référence. Cependant, la méthodologie possède des lacunes que nous nous proposons de combler tels que le test des applications uniquement du point de vue du terminal. La section suivante présente une amélioration de la méthodologie de Lancia pour le test d'applications de paiement en boîte noire.

4.5.2 Architecture de fuzzing dans WinSCard Tools

Comme vu précédemment, un fuzzer est composé des trois grands composants suivants :

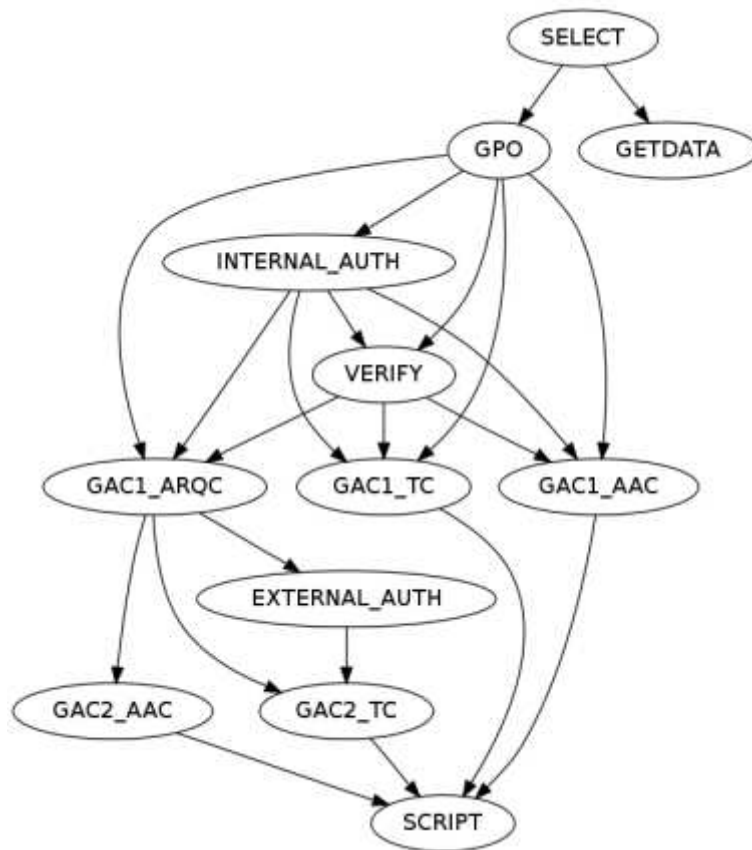


FIGURE 4.17 – Modèle du protocole EMV selon Lancia dans [157]

- génération de données,
- transmission de données,
- surveillance de la cible.

Nous présentons ici comment WinSCard Tools fournit déjà tous les éléments nécessaires à l’implémentation de ces trois couches et les objets de données qu’il manipule.

4.5.2.1 Les objets de données

La pile protocolaire régissant la communication entre l’applet et le monde extérieur est composée de trois couches définies dans le standard ISO 7816 [153] :

1. Couche physique
2. Couche transport
3. Couche application

La couche physique (ISO 7816-3) décrit la convention de codage des octets, la fréquence d'horloge, la vitesse de transmission, etc. Les APDUs, données du niveau applicatif, sont transmises au protocole de transmission. Les structures de données échangées à ce niveau sont appelées TPDU (*Transmission Protocol Data Unit*). Une applet reçoit les APDUs en entrée par le biais du système d'exploitation. Elles sont définies dans ISO 7816-4 et il en existe deux types :

- l'APDU de commande (C-APDU) ;
- l'APDU de réponse (R-APDU).

Une APDU de commande est toujours pairée à une APDU de réponse. Leurs structures sont illustrées respectivement dans 4.1 et 4.2.

En-tête obligatoire				Corps optionnel		
CLA	INS	P1	P2	Lc	Champ de données	Le

TABLE 4.1: Structure d'une C-APDU

Corps optionnel		En-queue obligatoire	
Champ de données		SW1	SW2

TABLE 4.2: Structure d'une R-APDU

La C-APDU est constituée de :

- un en-tête obligatoire de quatre octets :
 - **CLA**, l'octet de classe identifie la catégorie de l'APDU (exemple : 0x00 désigne une commande normalisée par l'ISO 7816-4, 0x80 désigne une commande propriétaire),
 - **INS**, l'octet d'instruction spécifie la commande à exécuter,
 - **P1** et **P2**, deux octets de paramètres spécifiques à l'instruction ;
- un corps optionnel de taille variable :
 - **Lc**, l'octet désignant la longueur du champ de données,
 - le **champ de données**,
 - **Le**, l'octet désignant la longueur des données attendues en retour.

La R-APDU est constituée de :

- un corps optionnel de taille variable contenant le **champ de données** de taille **Le** spécifiée dans la C-APDU ;

- SW1 et SW2, deux octets dits *Status Word* indiquant le statut d'exécution de la commande.

4.5.2.2 Génération de données

La génération des données est rendue plus aisée grâce à un système de dictionnaire. Ce dictionnaire recense les tags de la spécification, leurs types (binaire, date, chaîne de caractères, chiffres codés en BCD, ...)) et la librairie WSCT responsable de leur génération (cf. figure 4.18). Par exemple, pour générer le montant de la transaction (tag 9F02) de manière aléatoire mais conforme à la spécification, WSCT fera appel par réflexion à l'objet `BCDTLVObject` et invoquera la méthode `GenerateRandomValue()`. De plus, WSCT offre une API haut niveau implémentant des fonctions EMV. Par exemple, la méthode `ReadApplicationData()` envoie toutes les commandes READ RECORD nécessaires pour lire les informations présentes dans les enregistrements indiqués dans l'*Application File locator* (AFL).

4.5.2.3 Transmission de données

La transmission des commandes APDU vers un lecteur PC/SC est nativement présente dans WSCT. De plus, le modèle en couches utilisé dans WSCT permet de développer une couche d'adaptation permettant d'envoyer les commandes APDU vers un autre support qu'une carte insérée dans un lecteur PC/SC (par exemple vers un serveur IP, un simulateur...). Nous avons utilisé ce mécanisme pour envoyer les commandes en même temps à la carte et à l'implémentation de référence.

4.5.2.4 Surveillance de la cible

La surveillance de la cible est facilitée par l'utilisation d'événements au sein de WSCT. Ces événements sont déclenchés par l'envoi de commandes à la carte et la réception de données retournées par la carte. Ces événements sont capturés par le gestionnaire d'événements de notre plateforme qui compare les données retournées par la carte ciblée et un résultat attendu. Si les données sont différentes alors le gestionnaire lève une exception et stocke les différents éléments pour analyse.

Dans les sections suivantes, nous exposons deux méthodes de fuzzing pour carte à puce basé sur WinSCard Tools. La première est basée sur une évaluation de la réponse de la carte à une commande, la seconde est basée sur une implémentation de référence de l'application testée..


```

<tlvDesc hexaValue="9F49" name="DDOL" longName="Dynamic Data
    Authentication Data Object List" source="ICC">
    <value format="b" maxSize="252"/>
    <className>WSCT.EMV.Objects.DataObjectList</className>
    <dll>WSCT.EMV.dll</dll>
</tlvDesc>

<tlvDesc hexaValue="9F4D" name="LogEntry" longName="Log Entry"
    source="ICC">
    <value format="b" size="2"/>
    <className>WSCT.EMV.Objects.LogEntry</className>
    <dll>WSCT.EMV.dll</dll>
</tlvDesc>

<tlvDesc hexaValue="9F81" name="ConsCounter" longName="Consecutive
    Offline Transactions Number" source="ICC">
    <value format="b" size="6"/>
    <className>WSCT.Helpers.BasicEncodingRules.BinaryTLVObject
    </className>
</tlvDesc>

<tlvDesc hexaValue="9F02" name="AmountAuth" longName="Amount,
    Authorised" source="Terminal">
    <value format="n" size="12"/>
    <className>WSCT.Helpers.BasicEncodingRules.BCDTLVObject</className>
</tlvDesc>

```

FIGURE 4.18 – Exemples de descriptions de tags contenus dans le dictionnaire

4.5.3 1^{ère} approche : fuzzing sans implémentation de référence

De manière assez instinctive, il est nous est apparu qu'un framework de tests idéal serait un framework capable d'évaluer la qualité de la donnée envoyée à la carte en fonction de l'effet produit par rapport à un objectif initial. En d'autres termes, un framework capable de modifier automatiquement les APDUs envoyées en fonction de la réponse de la carte afin de découvrir des vulnérabilités. La méthode proposée utilise les algorithmes génétiques (AGs) pour générer les données envoyées à l'application et évaluer leur qualité.

4.5.3.1 Les algorithmes génétiques

Les algorithmes génétiques sont une méthode de recherche stochastique introduits dans les années 1970 par John Holland [167] et par Ingo Rechenberg [168]. Basés sur la simplification des mécanismes d'évolution naturelle, les algorithmes génétiques opèrent

sur une population de solutions plutôt qu'une seule et emploient des heuristiques telles que la sélection, le croisement et la mutation d'individus de la génération précédente pour obtenir une meilleure population [169]. Un algorithme génétique est défini en considérant cinq données essentielles [170] :

1. *la représentation de la solution ou génotype* : chaque individu d'une population est décrit par son génotype, i.e. un ensemble de caractéristiques qui le caractérisent,
2. *la population initiale* : elle comprend un ensemble d'individus décrits par leurs génotypes. Cette population peut être soit générée aléatoirement soit prédéfinie,
3. *la fonction d'aptitude* : elle mesure l'aptitude d'un individu en fonction de son génotype,
4. *les opérations sur les génotypes* : elles définissent les modifications apportées aux génotypes afin de faire évoluer une population au cours des générations. Il existe trois types d'opérations :
 - la mutation d'un individu : les gènes de l'individu sont légèrement modifiés afin de mieux s'adapter à l'environnement,
 - la sélection d'un individu : seuls les individus adaptés à l'environnement survivent à la génération suivante,
 - le croisement de deux individus : les gènes de l'individu résultant du croisement de deux individus est une combinaison des gènes de ses parents,
5. *critère d'arrêt de l'évolution de la population* : il met fin à l'évolution d'une population en fonction de l'aptitude du meilleur individu ou de nombre de générations produites.

La cinématique de l'exécution d'un algorithme génétique est la suivante :

1. définition de la population initiale et calcul de l'aptitude de chaque individu,
2. sélection et mutation des individus de la population courante,
3. croisement des individus restants de la population courante,
4. évaluation des individus de la population,
5. revenir à l'étape 2 si le critère d'arrêt n'est pas satisfait.

Pour implémenter les couches de génération de données et de surveillance de la cible de notre framework de fuzzing, nous avons utilisé cet algorithme génétique. Dans cet algorithme, nous avons introduit un critère d'évaluation que nous avons développé comme fonction d'aptitude. Ce critère quantifie la qualité de la commande `GENERATE AC` envoyée à l'application de paiement afin de provoquer et de détecter une vulnérabilité. Dans la section suivante, nous détaillons la définition de ce critère et précisons les données utilisées dans l'algorithme génétique.

4.5.3.2 Dispositif expérimental

Notre framework de test par les AGs est basé sur l'architecture décrite à la section 4.5.2. Nous ajoutons à la partie métier responsable du fuzzing, la librairie AForge.NET [171] offrant une implémentation d'AG open source. Nous installons et personnalisons sur une carte à puce une applet que nous avons développée. Dans cette applet, nous avons implémenté une partie des spécifications MasterCard M/Chip [172] dont la cinématique de transaction répond à celle présentée à la section 4.2. Nous avons allégé le développement en ne gardant que le strict nécessaire à la réalisation d'une transaction de paiement. Le tableau 4.3 résume les différents éléments de l'expérimentation.

Framework de fuzzing	WinSCard Tools
Langage du framework	C#
Carte à puce utilisée	Simulateur JCOP 2.4.1
Application de paiement	MasterCard MChip 4
Librairie d'AG	AForge .NET
Nombre d'individus de la population	10000
Nombre d'itérations	5000
Algorithme de sélection des meilleurs individus	Approche élitiste (on conserve la moitié des individus maximisant la fonction d'aptitude)
Méthode de croisement des individus	Permutation de deux gènes tirés aléatoirement
Méthode de mutation des individus	Combinaison de deux parties du génotype des parents tirés aléatoirement (illustré à la figure 4.19)
Données représentées par les individus	Données de la commande GENERATE AC (cf. Figure 4.20)
Fonction d'évaluation des individus	Attribution d'un score à la réponse au GENERATE AC (CID, CVR ...) (cf. 4.5.3.3)
Coefficient α multiplicateur du score	$\alpha = 1$ si le cryptogramme est un AAC $\alpha = 3$ si le cryptogramme est un ARQC $\alpha = 5$ si le cryptogramme est un TC

TABLE 4.3: Synthèse des éléments et paramètres de l'expérimentation

La section suivante décrit la mise en œuvre de notre approche sur cette applet.

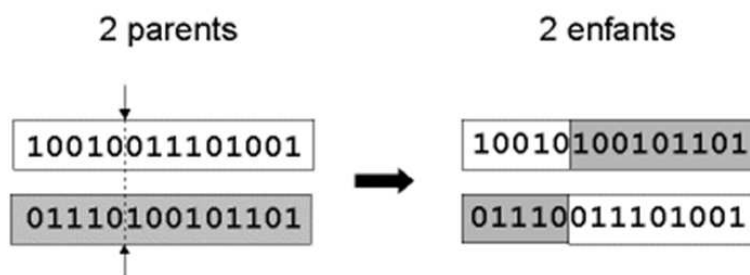


FIGURE 4.19 – Croisement de deux individus

4.5.3.3 Mode opératoire

Dans cette approche basée sur les AGs, nous nous appuyons sur la cinématique d'une transaction de paiement et utilisons les AGs pour la génération de données de la commande `GENERATE AC`. Ainsi, le génotype de nos individus représente les données de la commande `GENERATE AC`, *i.e.* les données relatives à l'élément de données `CDOL 1` ou `CDOL 2` selon le cas. Les données du `CDOL 1` de l'application MasterCard M/Chip est donnée à la figure 4.20. Le critère d'évaluation du résultat de la commande `GENERATE AC` est une combinaison du type de cryptogramme retourné par l'application indiqué par l'élément de donnée *Cryptogram Information Data* (`CID`) et les vérifications effectuées par la carte représentées dans l'élément de donnée *Card Verification Results* (`CVR`). Le `CID` et le `CVR` sont tous les deux retournés par la carte dans la réponse à la commande `GENERATE AC`.

Le `CVR` de l'application MasterCard M/Chip comprend six octets : les trois premiers sont utilisés à titre d'information et les trois derniers sont utilisés pour information et pour le processus de prise de décision. Nous détaillons à la figure 4.21 les bits utilisés pour le critère d'évaluation. Ainsi, le score d'évaluation est incrémenté de 1 si un des bits indiqués dans les octets 1 à 3 est à 0 et incrémenté également de 1 si un des bits indiqués dans les octets 4 à 6 est à 1. Ce score est ensuite multiplié par un coefficient α dont la valeur est fonction de la valeur du `CID`, *i.e.* du cryptogramme retourné : 1 si le cryptogramme est un AAC, 3 si le cryptogramme est un ARQC et 5 si le cryptogramme est un TC.

Du côté du framework de test, nous avons enrichi la librairie AForge.NET d'un nouvel objet de type `Chromosome`, capable de générer les données de la commande `GENERATE AC` à partir du `CDOL 1` ou du `CDOL 2` et d'effectuer les croisements et mutations. Nous avons également développé une fonction d'évaluation et une fonction d'aptitude dédiées à ce nouveau type de chromosome. Lors de cette campagne de fuzzing, nous nous étions fixé un objectif simple afin de valider l'approche : fuzzer la commande `GENERATE AC` responsable de l'acceptation de la transaction par

Data Element	Tag	Length
<i>Amount, Authorized (Numeric)</i>	'9F02'	6
<i>Amount, Other (Numeric)</i>	'9F03'	6
<i>Terminal Country Code</i>	'9F1A'	2
<i>Terminal Verification Results</i>	'95'	5
<i>Transaction Currency Code</i>	'5F2A'	2
<i>Transaction Date</i>	'9A'	3
<i>Transaction Type</i>	'9C'	1
<i>Unpredictable Number</i>	'9F37'	4
<i>Terminal Type</i>	'9F35'	1
<i>Data Authentication Code</i>	'9F45'	2
<i>ICC Dynamic Number</i>	'9F4C'	8
<i>CVM Results</i>	'9F34'	3

FIGURE 4.20 – Données du CDOL 1 de l'application MasterCard M/Chip

l'application afin de tester notre implémentation. Le mode opératoire est le suivant :

1. Définition des coefficients utilisés pour le calcul du score : $\alpha = 1$, $\beta = 3$, $\gamma = 5$.
2. Création d'une population de n chromosomes.
3. Pour chaque itération i , effectuer une transaction de paiement pour les n individus :
 - a) envoi des commandes **SELECT**, **GET PROCESSING OPTIONS**, **READ RECORD** et **GET DATA** « par défaut »,
 - b) aléatoirement, envoi du code PIN avec la commande **VERIFY** (nécessite la connaissance préalable du code PIN),
 - c) la première commande **GENERATE AC** est envoyée en demandant un TC avec les données du chromosome relatives au CDOL 1. Si un ARQC est retourné par l'application, la seconde commande **GENERATE AC** est envoyée en demandant un TC avec les données du chromosome relatives au CDOL 2,
 - d) basé sur les données renvoyées par l'application, chaque individu est évalué par le critère d'évaluation détaillé plus haut,
 - e) le meilleur individu est sélectionné pour la production de la génération suivante.

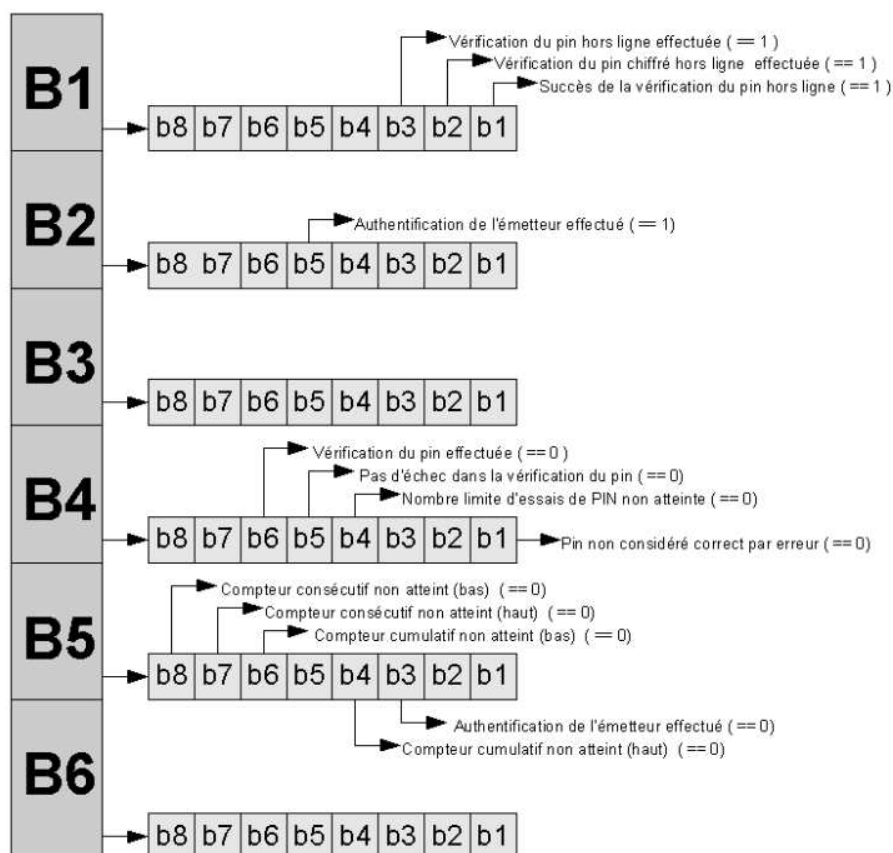


FIGURE 4.21 – Données du Card Verification Results de l'application Master-Card M/Chip utilisés pour le critère d'évaluation

Toutes les transactions sont sauvegardées et lorsqu'une remise à zéro des compteurs *offline* est détectée ou qu'une transaction est acceptée au-delà du nombre de transactions *offline* consécutives autorisées, cette transaction est également sauvegardée dans le log des « anomalies ». La remise à zéro des compteurs est facilement détectable car l'application les retourne dans la réponse à la commande GENERATE AC dans l'élément de données *Issuer Application Data*.

4.5.3.4 Résultats et discussion

La première constatation est que les cartes utilisées pour réaliser nos sessions de fuzzing sont inadaptées à ce genre de pratique. En effet, nous avons rendus totalement inutilisables deux cartes de suite après approximativement vingt-cinq mille transactions chacune. Nous avons alors décidé de ne plus réaliser nos tests sur de réelles cartes mais de charger les applications dans un simulateur (*cf.* figure 4.22). Ainsi, après nous être procuré une licence de l'outil NXP JCOP, nous l'avons intégré

```

file:///C:/Documents and Settings/valimi/My Documents/Visual Studio 2010/Projects/WinSCard Tools [git]/WSCT Fuzzing/WSCT.Fuzzing.EMV.ConsoleTests/bin/...
Status: No Error
> 00 CA 9F 83 00
<627454 nsec>
<= 9F 83 06 95 00 03 04 04 00 90 00
Status: No Error
Individual score: 40
Transaction #38 Overall duration: 0d:0h:0m:2s.968
> 00 A4 04 00 07 A0 00 00 00 04 10 10 00
<2894 usec>
<= 6F 17 04 07 A0 00 00 00 04 10 10 A5 0C 50 0A 4D
61 73 74 65 72 43 61 72 64 90 00
asterCard..
Status: No Error
> 00 A8 00 00 02 83 00 00
<3141 usec>
<= 77 16 02 02 79 00 94 10 10 02 02 01 18 01 01 00
20 01 01 00 20 01 02 00 90 00
w..y.....
Status: No Error
> 00 20 00 80 00 24 12 34 FF FF FF FF FF
<1863 usec>
<= 90 00
...$.4....
Status: No Error
> 00 AE 00 00 2B 57 04 23 85 70 36 86 95 83 97 30
80 32 17 E6 ED AD AD 29 53 41 43 09 17 8D 5A 9F
3C 12 EA DE C6 04 0D F1 FC 00 0E EA B3 AB DE 90
00
...+V.#.x6...0
-2.....>SAC...Z.
<= 90 00
<4458 usec>
<= 77 81 29 9F 27 01 80 9F 36 02 00 27 9F 10 12 01
10 A5 00 03 04 04 00 DE C5 00 00 00 00 00 0C
FF 9F 26 08 59 54 84 ED C8 04 8E 35 90 00
w..'.6..'....
..&.YT.....5...
Status: No Error
> 00 AE 40 00 1D FC 22 5B 29 51 A1 22 12 4F 1D 7C
83 00 08 00 8C 28 68 43 C0 4A B0 80 79 16 EA
6C 9F 00
...0..."\|>q"...0.!
.....<hC.J..y..
1..
<5444 usec>
<= 77 81 29 9F 27 01 00 9F 36 02 00 27 9F 10 12 01
10 25 10 03 04 04 00 DE C5 00 00 00 00 00 0C
FF 9F 26 08 48 C8 E7 00 B2 F8 01 F0 90 00
w..'.6..'....
%......
..&.H.....
Status: No Error
> 00 CA 9F 81 00
<509842 nsec>
<= 9F 81 01 0C 90 00
Status: No Error
> 00 CA 9F 82 00
<537499 nsec>
<= 9F 82 06 00 00 00 00 00 90 00
Status: No Error
Individual score: 0
Transaction #39 Overall duration: 0d:0h:0m:3s.0
> 00 A4 04 00 07 A0 00 00 00 04 10 10 00
<3103 usec>

```

FIGURE 4.22 – Utilisation du simulateur JCOF pour réaliser les sessions de fuzzing

à WSCT. Pour cela, nous avons utilisé l'exécutable du simulateur (`jcop.exe`) et avons porté l'API *offcard* JCOF vers C# de la même manière que nous l'avons fait pour Jena (cf. section 3.4.4.2). Ensuite, nous avons développé une couche d'adaptation permettant, au travers de cette couche, d'envoyer et recevoir des APDUs de manière transparente pour les couches supérieures. Certes, cette « dématérialisation » induit un biais car des vulnérabilités pourrait avoir une origine matérielle, mais cela nous a permis d'augmenter de manière considérable la vitesse d'exécution du framework. Par exemple, nous avons réussi à atteindre une vitesse de cinq mille transactions de paiement EMV par minute sur un ordinateur extrêmement puissant.

Après plusieurs sessions de fuzzing, à raison de plusieurs dizaines d'heures et millions de transactions, l'approche proposée ne nous a pas permis d'observer de remise à zéro des compteurs sur notre implémentation, mais nous avons néanmoins pu observer l'acceptation de plusieurs transactions *offline* au-delà du nombre de transactions *offline* consécutives autorisées. Cette information nous signale donc une anomalie dans notre implémentation de la spécification MasterCard ce qui est bien le but de notre framework de test. Ces transactions « illégitimes » sont sauvegardées et nous disposons de toutes les informations sur les données ayant provoqué la vulnérabilité détectée. Cependant, il est difficile de remonter à la séquence d'évènements ayant conduits à cette anomalie. Par exemple, nous avons détecté lors d'une campagne de fuzzing que des transactions arrivaient à être autorisées *offline*

au-delà de la 10000^{me} transaction comme le montre le listing 4.1. Dans ces conditions, il est très difficile de connaître exactement le contexte de l'anomalie et comment les transactions précédentes ont pu influencer. Nous avons alors opté pour une approche basée sur une implémentation de référence décrite dans la section suivante.

```

Transaction # 9670
1st GENERATE AC
>> 80 AE 40 00 2B 47 90 87 71 63 91 34 72 42 90 09 67 34 41 9D C4 B3 40 8F 18 10
    89 03 04 B9 FA EB 06 E8 36 9E 1B 78 D1 5C 01 C1 C5 8E 45 95 9F 16 00
    9F02 : 47 90 87 71 63 91 Amount, Authorised
    9F03 : 34 72 42 90 09 67 Amount, Other
    9F1A : 34 41 Terminal Country Code
    95 : 9D C4 B3 40 8F Terminal Verification Results
    5F2A : 18 10 Transaction Currency Code
    9A : 89 03 04 Transaction Date
    9C : B9 Transaction Type
    9F37 : FA EB 06 E8 Unpredictable Number
    9F35 : 36 Terminal Type
    9F45 : 9E 1B Data Authentication Code
    9F4C : 78 D1 5C 01 C1 C5 8E 45 ICC Dynamic Number
    9F34 : 95 9F 16 CVM Results
<< 77 81 29 9F 27 01 40 9F 36 02 16 BF 9F 10 12 01 10 95 00 03 04 84 00 9E 1B 00
    00 00 00 00 00 26 FF 9F 26 08 08 C0 59 42 20 02 84 0A 90-00
    9F27 : 40 Cryptogram Information Data
    9F36 : 16 BF Application Transaction Counter
    9F10 : 01 10 95 00 03 04 84 00 9E 1B 00 00 00 00 00 00 26 FF Issuer
        Application Data
    9F26 : 08 C0 59 42 20 02 84 0A Application Cryptogram
Card Verification Results
No 2nd Generate AC | TC returned in 1st Generate AC | Offline PIN verification
performed | Offline Encrypted PIN verification not performed | Offline PIN
verification successful | DDA not returned | CDA not returned in 1st Generate
AC | CDA not returned in 2nd Generate AC | Issuer Authentication not
performed | No CIAC-Default skipped on CAT3 | Right nibble of Script Counter:
0x00 | Right nibble of PIN Try Counter: 0x03 | Unable to go Online not
indicated | Offline PIN verification not performed | No failure of Offline
PIN verification | PTL not exceeded | International transaction |
International transaction | Terminal does not erroneously consider offline
PIN OK | LCOL exceeded | UCOL not exceeded | LCOTA not exceeded | UCOTA not
exceeded | Go Online on next transaction was not set | Issuer Authentication
Failed | No script received | No script failed | No Match Found in Additional
Table Check | Match Found in Additional Table Check |

Transaction # 11463
1st GENERATE AC
>> 80 AE 40 00 2B 81 74 86 85 85 75 27 56 70 83 91 12 32 51 23 A0 B1 FB 90 31 35
    78 08 23 3A 4B FD 20 5E 26 30 77 0E 8F 8E F6 18 94 A7 B4 07 0F 5F 00
    9F02 : 81 74 86 85 85 75 Amount, Authorised
    9F03 : 27 56 70 83 91 12 Amount, Other
    9F1A : 32 51 Terminal Country Code
    95 : 23 A0 B1 FB 90 Terminal Verification Results
    5F2A : 31 35 Transaction Currency Code
    9A : 78 08 23 Transaction Date
    9C : 3A Transaction Type
    9F37 : 4B FD 20 5E Unpredictable Number

```



```

9F35 : 26 Terminal Type
9F45 : 30 77 Data Authentication Code
9F4C : 0E 8F 8E F6 18 94 A7 B4 ICC Dynamic Number
9F34 : 07 0F 5F CVM Results
<< 77 81 29 9F 27 01 40 9F 36 02 1B 03 9F 10 12 01 10 95 08 03 04 84 00 30 77 00
00 00 00 00 00 2A FF 9F 26 08 14 5D 80 80 29 3C 66 EF 90-00
9F27 : 40 Cryptogram Information Data
9F36 : 1B 03 Application Transaction Counter
9F10 : 01 10 95 08 03 04 84 00 30 77 00 00 00 00 00 2A FF Issuer
Application Data
9F26 : 14 5D 80 80 29 3C 66 EF Application Cryptogram
Card Verification Results
No 2nd Generate AC | TC returned in 1st Generate AC | Offline PIN verification
performed | Offline Encrypted PIN verification not performed | Offline PIN
verification successful | DDA not returned | CDA not returned in 1st Generate
AC | CDA not returned in 2nd Generate AC | Issuer Authentication not
performed | CIAC-Default skipped on CAT3 | Right nibble of Script Counter:
0x00 | Right nibble of PIN Try Counter: 0x03 | Unable to go Online not
indicated | Offline PIN verification not performed | No failure of Offline
PIN verification | PTL not exceeded | International transaction |
International transaction | Terminal does not erroneously consider offline
PIN OK | LCOL exceeded | UCOL not exceeded | LCOTA not exceeded | UCOTA not
exceeded | Go Online on next transaction was not set | Issuer Authentication
Failed | No script received | No script failed | No Match Found in Additional
Table Check | Match Found in Additional Table Check |

```

Listing 4.1: Extrait des données sauvegardées montrant des transactions autorisées *offline* (le CID vaut 40)

4.5.4 2^{nde} approche : fuzzing avec une implémentation de référence

Dans [157], Lancia propose une approche basée sur un générateur de données par bloc et une surveillance de la cible par le biais d'une implémentation de référence. Dans cette section, nous proposons d'utiliser cette approche en l'adaptant à WSCT et d'améliorer les points qui ne nous semblent pas optimaux.

Dans la méthode proposée par Lancia, l'application cible est testée en suivant la cinématique d'une transaction d'un point de vue du terminal comme illustré à la figure 4.17. Cette méthode s'est montrée certes efficace, mais propose de tester une application uniquement dans le contexte d'une transaction. Une application de paiement est une machine d'état qui accepte les commandes entrantes selon une matrice d'acceptation telle qu'illustrée à la figure 4.23.

En testant l'application selon une cinématique statique, l'application n'est pas testée « en profondeur », seule sa conformité à la cinématique de paiement du point de vue du terminal l'est. Par exemple, la robustesse de la machine d'état n'est pas testée, *i.e.* le respect de la matrice d'acceptation est-il bien implémenté? L'approche

	IDLE	SELECTED	INITIATED
SELECT	Accept	Accept	Accept
GET PROCESSING OPTIONS	Reject	Accept	Reject
READ RECORD	Reject	Accept	Accept
COMPUTE CRYPTOGRAPHIC CHECKSUM	Reject	Reject	Accept

FIGURE 4.23 – Matrice d’acceptation de l’application MasterCard Paypass Magstripe (source : [173])

proposée consiste à réaliser un framework de test de l’application d’un point de vue carte et non plus terminal comme proposé par Lancia. Ainsi, l’application n’est plus testée de façon nominale, transactionnelle mais est testée du point de vue de l’implémentation stricte de la spécification. Pour cela, nous nous appuyons sur WSCT et nous testons les aspects suivants :

- respect de la machine d’état (acceptation des commandes et transitions)
- implémentation des commandes conformément à la spécification
- détection de vulnérabilités sécuritaires

4.5.4.1 Méthode proposée

L’architecture du framework de fuzzing est représentée à la figure 4.24. Dans cette architecture, les commandes générées par le fuzzer sont envoyées simultanément à l’application cible et à une implémentation de référence de cette application. Les réponses respectives de l’application cible et de l’implémentation de référence remontent au Comparateur/Gestionnaire d’évènements qui s’assure de l’égalité des deux réponses, *i.e.* que l’application cible a eu le comportement attendu. Dans le cas contraire, une requête est envoyée à l’implémentation de référence afin qu’elle fournisse une image de tous ses indicateurs internes. Ces indicateurs sont consignés pour analyse ultérieure.

La génération des données est effectuée de manière à tester l’application en profondeur comme l’explique en détail la section 4.5.4.3. Les commandes sont testées une par une, *i.e.* lorsqu’une commande est testée, nous envoyons une valeur prédéfinie pour les autres commandes. Des paramètres permettent de définir le nombre de commandes envoyées à chaque état de l’application avant de passer à l’état suivant.

4.5.4.2 Dispositif expérimental

Afin de valider notre méthodologie, nous avons développé l’application carte de la spécification MasterCard ainsi que son implémentation de référence pour notre

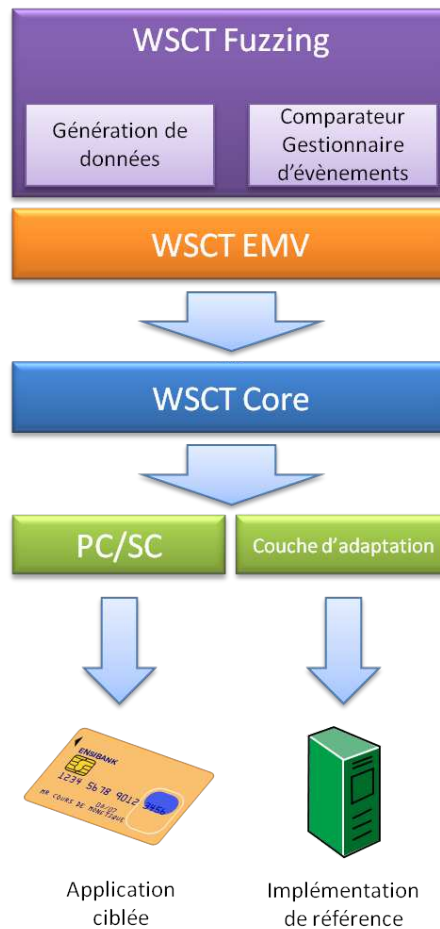


FIGURE 4.24 – Architecture du framework de fuzzing basée sur une implémentation de référence

framework de fuzzing. L'implémentation de référence consiste en une classe écrite dans le même langage que WSCT, *i.e.* C#. Elle prend en entrée une APDU de commande (objet `CommandAPDU`) et renvoie une APDU de réponse (objet `ResponseAPDU`). Ainsi, son utilisation est transparente à *WSCT Core*, qui par le biais de la méthode `transmit()` transmet une `CommandAPDU` et renvoie une `ResponseAPDU`. Elle comprend principalement une méthode `Process()` qui, de la même façon qu'une applet Java Card, dispatche les commandes vers une méthode dédiée à son traitement en fonction de l'octet d'instruction. Par exemple, une commande `READ RECORD` sera envoyée à la méthode `Process(CommandAPDU apdu)` qui la dispatche vers la méthode `ProcessReadRecord(CommandAPDU apdu)`. Par manque de temps, nous avons choisi de ne pas prendre une application trop complexe à développer ni du côté carte ni du côté framework. Nous avons donc choisi d'implémenter la spécification MasterCard Paypass Magstripe [173], moins complexe que la spécification EMV complète

MasterCard M/Chip [172]. Paypass Magstripe est une application *online*, elle renvoie des données qui doivent être transmises à l'émetteur pour autorisation à chaque transaction. Elle reconnaît les commandes listées ci-dessous et suit la cinématique de transaction décrite à la figure 4.25.

- SELECT
- GET PROCESSING OPTIONS
- READ RECORD
- COMPUTE CRYPTOGRAPHIC CHECKSUM

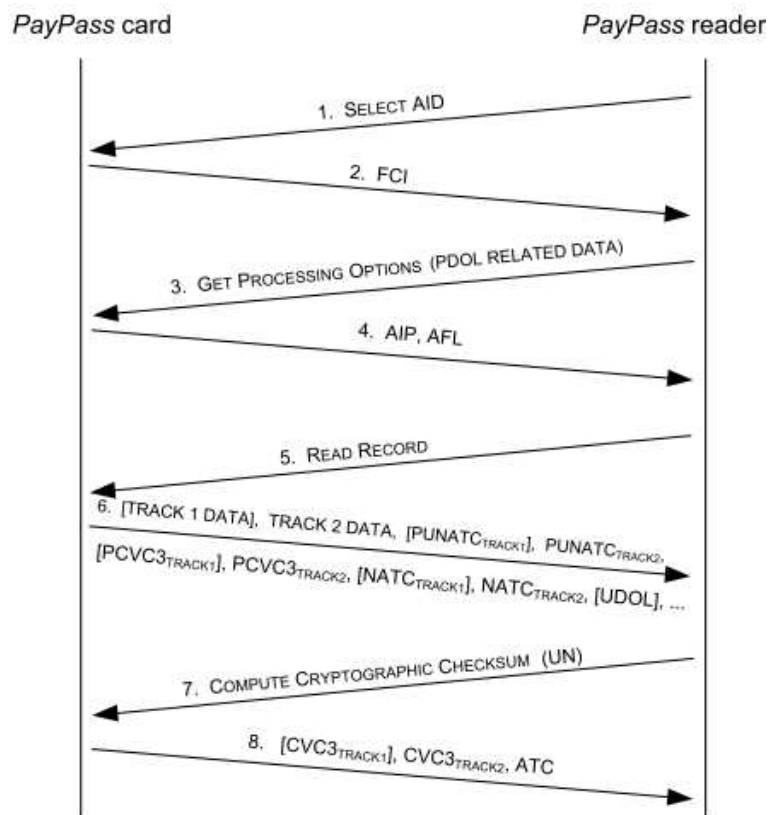


FIGURE 4.25 – Cinématique d’une transaction MasterCard Paypass Magstripe

La matrice d’acceptation des commandes est illustrée à la figure 4.23. Les états de la machine d’état sont les suivants :

- IDLE
- SELECTED
- INITIATED

Le diagramme UML du framework de fuzzing proposé est représenté à la figure 4.26. On y distingue la partie « Core » et les implémentations spécifiques pour notre cas. La

partie Core comprend des *Helpers* proposant des méthodes utiles telle que la génération de données en fonction du tag reconnu depuis le dictionnaire, un dictionnaire, une machine d'état (`IStateMachine`), un objet `EMVFuzzingApplication` faisant l'interface avec la carte et l'implémentation de référence et un objet `EMVFuzzer` comprenant la logique métier de notre framework. L'implémentation ad hoc, dans notre cas, est composée de la machine d'état Paypass Magstripe (`MasterCardStateMachine`), un dictionnaire décrivant les éléments de données MasterCard (`Dictionnaire.EMVTag.MasterCard.xml`) et des méthodes utiles (`MasterCardUtilities`). Le tableau 4.4 résume les différents éléments de l'expérimentation.

Framework de fuzzing	WinSCard Tools
Langage du framework et de l'implémentation de référence	C#
Carte à puce utilisée	Simulateur JCOP 2.4.1
Application de paiement	MasterCard Paypass Magstripe 3.3
Paramètres du fuzzer	α : nombre de commandes envoyées dans l'état SELECTED (ici $\alpha = 2$) β : nombre de commandes envoyées dans l'état INITIATED (ici $\beta = 2$) γ : nombre de commandes envoyées dans l'état ONLINE (ici $\gamma = 3$) δ : nombre de commandes envoyées dans l'état SCRIPTS (ici $\delta = 0$) $\lambda = 500$, nombre de transactions de fuzzing à réaliser sur chaque commande avant de passer à une autre
Matrice du fuzzer	Matrice d'acceptation des commandes de l'application de paiement

TABLE 4.4: Synthèse des éléments et paramètres de l'expérimentation

4.5.4.3 Mode opératoire

Pour tester la machine d'état et l'implémentation des commandes, nous nous appuyons sur la cinématique d'une transaction car elle fait passer l'application dans les différents états de la machine d'état jusqu'à la fin de la transaction. A chaque état, nous testons les éléments suivants :

- l'application accepte-t-elle les commandes « légitimes » ?
- l'application refuse-t-elle les commandes « illégitimes » ?
- l'application renvoie-t-elle au monde extérieur les données dites « publiques » (commandes `READ RECORD` et `GET DATA`)

- l'application renvoie-t-elle au monde extérieur uniquement les données dites « publiques » (commandes **READ RECORD** et **GET DATA**)

Afin de tester l'acceptation des commandes légitimes et illégitimes, nous pourrions envoyer dans chaque état toutes les commandes reconnues par l'application, mais cette méthode présente une combinatoire très importante. Pour réduire la combinatoire, nous définissons un paramètre fixant le nombre de commandes envoyées dans chaque état. Les commandes envoyées sont alors tirées au hasard à chaque itération, *i.e.* à chaque transaction. Pour rendre cette approche flexible, nous définissons un paramètre pour chaque état :

- α : nombre de commandes envoyées dans l'état **SELECTED**
- β : nombre de commandes envoyées dans l'état **INITIATED**
- γ : nombre de commandes envoyées dans l'état **ONLINE**
- δ : nombre de commandes envoyées dans l'état **SCRIPTS**

Nous n'avons pas défini de paramètre pour l'état `IDLE`, car seule la commande `SELECT` peut sortir les applications de cet état. En plus de ces paramètres, nous pointons la commande permettant de faire passer l'application d'un état à un autre. Par exemple, la commande `GET PROCESSING OPTIONS` fait passer les applications dans l'état `INITIATED`, la commande `GENERATE AC` fait passer les applications dans l'état `ONLINE` lorsque celle-ci génère un `ARQC`.

En plus de ces tests d'acceptation de commandes visant à « stresser » la machine d'état, le framework réalise le fuzzing d'une commande en particulier par transaction. Un paramètre λ indique le nombre de transactions de fuzzing à réaliser sur chaque commande avant de passer à une autre. Par exemple, on fera varier les paramètres de la commande `READ RECORD` pendant 500 transactions puis sur la commande `GET PROCESSING OPTIONS`. Lors des tests fonctionnels décrits ci-dessus, si une commande est désignée comme celle sur laquelle se porte le fuzzing alors le framework la génère en utilisant sa connaissance du protocole et de cette commande en particulier. Sinon, le framework envoie la commande avec une valeur par défaut pré-définie. Les tests d'acceptation des commandes sont réalisés de la manière suivante :

1. l'application passe de l'état `IDLE` à l'état `SELECTED` en recevant la commande `SELECT`,
2. à l'état `SELECTED`, le fuzzer envoie α commandes puis envoie la commande provoquant la transition à l'état `INITIATED`,
3. à l'état `INITIATED`, le fuzzer envoie β commandes puis envoie la commande provoquant la transition à l'état `ONLINE` ou `SCRIPTS`,
4. etc.

Lorsque la réponse de la carte et de l'implémentation de référence diffèrent, le framework enregistre les informations de la transaction en cours :

- le numéro de la transaction,
- l'historique des commandes depuis l'envoi de la commande `SELECT`,
- la réponse renvoyée par la carte à la dernière commande,
- la réponse renvoyée par l'implémentation de référence,
- l'état des indicateurs internes de l'implémentation de référence.

L'implémentation s'appuie sur une matrice M symbolisant la matrice d'acceptation des commandes. La dimension 1 représente les états et la dimension 2 représente les commandes. L'élément $M[état, commande]$ représente l'acceptation ou non de la commande dans l'état donné. L'algorithme des tests fonctionnels est donné à la figure « Algorithme 1 ».

L'utilisation d'une implémentation de référence permet également la découverte de vulnérabilités sécuritaires telles que la remise à zéro des compteurs offline. En effet, en faisant une comparaison entre l'état des compteurs de la carte et de l'implémentation de référence, il est possible de déterminer si, comme dans [157], une combinaison de données sur l'envoi du 1^{er} et du 2nd **GENERATE AC** a remis les compteurs offline à zéro. Cette vulnérabilité s'avère critique car elle annihile la gestion du risque mise en place par l'émetteur. La récupération de l'état des compteurs peut se faire soit par la commande **GET DATA** pour récupérer et faire la soustraction entre les éléments de données *Application Transaction Counter* (ATC) et Last Online ATC Register ; soit ils sont « remontés » directement dans les *Issuer Application Data* (IAD) lors de la réponse à la commande **GENERATE AC** (cela dépend des spécifications).

```

 $\alpha, \beta, \gamma, \delta, \lambda$  : entier
compteurFuzzing : entier
accepteCommande : booléen
etat :  $\in$  {IDLE, SELECTED, INITIATED, ONLINE, SCRIPTS}
commande :  $\in$  {SELECT, GET PROCESSING OPTIONS, READ RECORD, ...}
commandeFuzzing :  $\in$  {SELECT, GET PROCESSING OPTIONS, READ RECORD, ...}
M[etat, commande] : matrice d'états et de commandes
acceptation :  $\in$  {ACCEPTEE, REJETEE}

etat  $\leftarrow$  IDLE
commande  $\leftarrow$  SELECT
EnvoyerCommande(commande)

Pour etat de SELECTED à SCRIPTS faire
  param : entier
  Selon que
    etat = SELECTED : param  $\leftarrow$   $\alpha$ 
    etat = INITIATED : param  $\leftarrow$   $\beta$ 
    etat = ONLINE : param  $\leftarrow$   $\gamma$ 
    etat = SCRIPTS : param  $\leftarrow$   $\delta$ 
  Fin Selon que
  Pour i de 1 à param faire
    Si (commande = commandeFuzzing) Alors
      commande  $\leftarrow$  GenererCommandeFuzzing(commande)
      compteurFuzzing  $\leftarrow$  compteurFuzzing + 1
      Si (compteurFuzzing =  $\lambda$ ) Alors
        commandeFuzzing  $\leftarrow$  ProchaineCommandeFuzzing()
        compteurFuzzing  $\leftarrow$  0
      Fin Si
    Sinon
      commande  $\leftarrow$  GenererCommandeParDefaut(commande)
    Fin Si
  Fin Pour
Fin Pour

```

Algorithme 1: Algorithme du test de la machine d'état et de l'implémentation des commandes

```

Pour etat de SELECTED à SCRIPTS faire
    [Suite]
    Pour i de 1 à param faire
        [Suite]
        Si ( $M[etat, commande] = ACCEPTEE$ ) Alors
            |  $accepteCommande \leftarrow \text{vrai}$ 
        Sinon
            |  $accepteCommande \leftarrow \text{faux}$ 
        Fin Si
         $EnvoyerCommande(commande)$ 
    Fin Pour

     $commande \leftarrow CommandeTransition(etat)$ 
     $EnvoyerCommande(commande)$ 
     $TransitionEtat(etat)$ 
Fin Pour

Fonction  $EnvoyerCommande(commande)$  :
     $reponseCarte$  : tableau d'octets
     $reponseIR$  : tableau d'octets

     $reponseCarte \leftarrow EnvoyerCommandeVersCarte(commande)$ 
     $reponseIR \leftarrow EnvoyerCommandeVersIR(commande)$ 
    Si ( $reponseCarte \neq reponseIR$ ) Alors
        |  $LoguerTransaction()$ 
    Fin Si
Fin

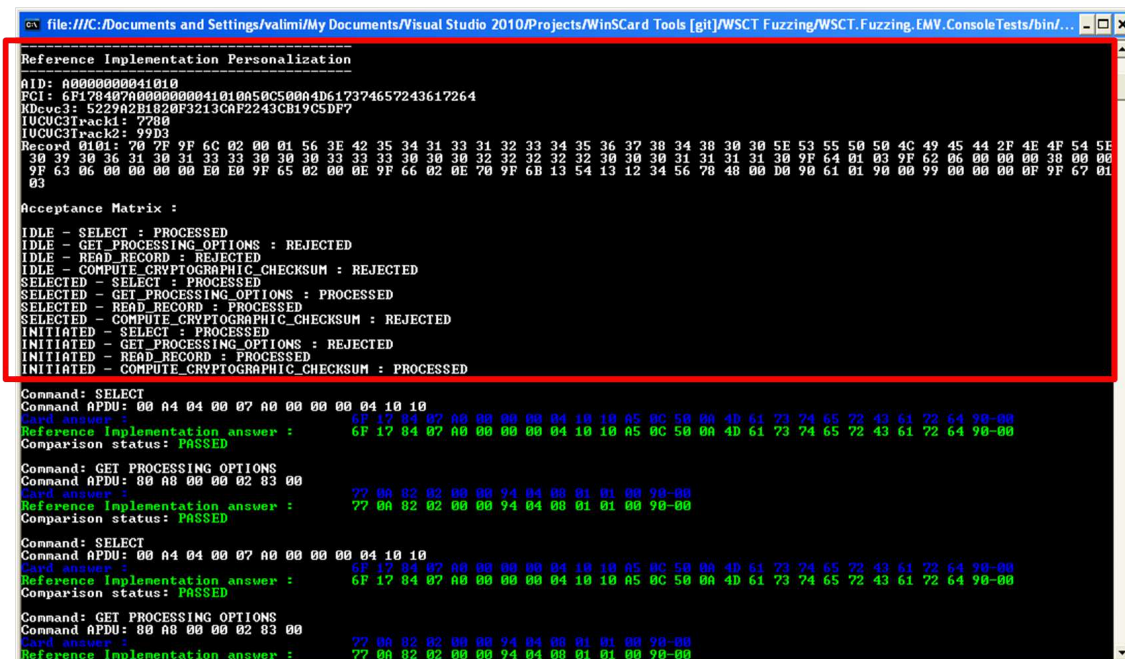
```

Algorithme 2: Algorithme du test de la machine d'état et de l'implémentation des commandes (suite)

4.5.4.4 Résultats et discussion

Comme le montre la figure 4.27, nous avons mis en œuvre l'implémentation de référence en créant sa matrice d'acceptation des commandes et la personnalisant avec les mêmes données que l'application cible. Nous avons ensuite exécuté notre algorithme sur quelques milliers d'itérations. De la même manière que pour l'approche basée sur les AGs, nous avons utilisé le simulateur JCOP et y avons chargé l'application cible.

Sur l'application MasterCard Paypass Magstripe, notre framework de fuzzing n'a



```

Reference Implementation Personalization
AID: A0000000041010
PCI: 61784B7A0000000041010A50C500A4D617374657243617264
KDCvc3: 5229A2B1820F3213CAF2243CB19C5DF7
IUCUC3Track1: 7780
IUCUC3Track2: 99D3
Record B101: 70 7F 6C 02 00 01 56 3E 42 35 34 31 33 31 32 33 34 35 36 37 38 34 38 30 30 5E 53 55 50 50 4C 49 45 44 2F 4E 4F 54 5E
30 39 30 36 31 30 31 33 33 30 30 33 33 33 30 30 32 32 32 32 30 30 31 31 31 31 30 9F 64 01 03 9F 62 06 00 00 00 38 00 06
9F 63 06 00 00 00 00 E0 E0 9F 65 02 00 0E 9F 66 02 0E 70 9F 6B 13 54 13 12 34 56 78 48 00 D0 90 61 01 90 00 99 00 00 00 0F 9F 67 01
03
Acceptance Matrix :
IDLE - SELECT : PROCESSED
IDLE - GET_PROCESSING_OPTIONS : REJECTED
IDLE - READ_RECORD : REJECTED
IDLE - COMPUTE_CRYPTOGRAFIC_CHECKSUM : REJECTED
SELECTED - SELECT : PROCESSED
SELECTED - GET_PROCESSING_OPTIONS : PROCESSED
SELECTED - READ_RECORD : PROCESSED
SELECTED - COMPUTE_CRYPTOGRAFIC_CHECKSUM : REJECTED
INITIATED - SELECT : PROCESSED
INITIATED - GET_PROCESSING_OPTIONS : REJECTED
INITIATED - READ_RECORD : PROCESSED
INITIATED - COMPUTE_CRYPTOGRAFIC_CHECKSUM : PROCESSED

Command: SELECT
Command APDU: 00 A4 04 00 07 A0 00 00 00 04 10 10
Card answer : 6F 17 84 07 A0 00 00 00 04 10 10 A5 0C 50 00 4D 61 73 74 65 72 43 61 72 64 90 00
Reference Implementation answer : 6F 17 84 07 A0 00 00 00 04 10 10 A5 0C 50 00 4D 61 73 74 65 72 43 61 72 64 90 00
Comparison status: PASSED

Command: GET PROCESSING OPTIONS
Command APDU: 80 A8 00 00 02 83 00
Card answer : 77 00 82 02 00 00 94 04 00 01 01 00 90 00
Reference Implementation answer : 77 00 82 02 00 00 94 04 00 01 01 00 90 00
Comparison status: PASSED

Command: SELECT
Command APDU: 00 A4 04 00 07 A0 00 00 00 04 10 10
Card answer : 6F 17 84 07 A0 00 00 00 04 10 10 A5 0C 50 00 4D 61 73 74 65 72 43 61 72 64 90 00
Reference Implementation answer : 6F 17 84 07 A0 00 00 00 04 10 10 A5 0C 50 00 4D 61 73 74 65 72 43 61 72 64 90 00
Comparison status: PASSED

Command: GET PROCESSING OPTIONS
Command APDU: 80 A8 00 00 02 83 00
Card answer : 77 00 82 02 00 00 94 04 00 01 01 00 90 00
Reference Implementation answer : 77 00 82 02 00 00 94 04 00 01 01 00 90 00

```

FIGURE 4.27 – Mise en œuvre de l’implémentation de référence : personnalisation et construction de la matrice d’acceptation des commandes

pas pu mettre en évidence de vulnérabilité sécuritaire. En effet, cette application est dite *online*, *i.e.* tous les éléments y compris les cryptogrammes sont vérifiés par la banque et il n’existe pas de mécanisme de gestion de risque *offline*.

En revanche, durant la campagne de tests, des vulnérabilités fonctionnelles ont été détectées nous aidant ainsi à corriger notre implémentation. Ces non-conformités sont au nombre de trois et sont détaillées ci-après.

Tout d’abord, les tests ont révélé une implémentation non conforme de la commande GET PROCESSING OPTIONS comme illustrée à la figure 4.28. A un moment de l’algorithme de test, la commande SELECT est succédée de trois commandes GET PROCESSING OPTIONS successives. L’erreur d’implémentation dans l’applet intervient sur la troisième répétition. Si on se réfère à la matrice d’acceptation des commandes de l’application cible et à la matrice des états de destination de la commande GET PROCESSING OPTIONS présentée à la figure 4.29, l’application est dans l’état INITIATED après la première occurrence, refuse la deuxième, renvoie ‘6985’, passe à l’état SELECTED puis accepte la troisième occurrence. L’application cible refuse cette troisième occurrence de la commande GET PROCESSING OPTIONS alors que l’implémentation de référence l’accepte conformément à la spécification.

Ensuite, nous avons détecté une non-conformité sur la commande GET DATA. Celle-ci, selon la spécification Mastercard, ne devrait être acceptée que pendant la

```

file:///C:/Documents and Settings/valimi/My Documents/Visual Studio 2010/Projects/WinSCard Tools [git]/WSCT Fuzzing/WSCT.Fuzzing.EMV.Con...
Reference Implementation answer : 6A-86
Comparison status: PASSED

Command: READ RECORD
Command APDU: 00 B2 B5 07 00
Card answer : 6A-86
Reference Implementation answer : 6A-86
Comparison status: PASSED

Command: SELECT
Command APDU: 00 A4 04 00 07 A0 00 00 00 04 10 10
Card answer : 6F 17 84 07 A0 00 00 00 04 10 10 A5 0C 50 0A 4D 61 73 74 65 72 43 61 72 64 90-00
Reference Implementation answer : 6F 17 84 07 A0 00 00 00 04 10 10 A5 0C 50 0A 4D 61 73 74 65 72 43 61 72 64 90-00
Comparison status: PASSED

Command: COMPUTE CRYPTOGRAPHIC CHECKSUM
Command APDU: 00 2A 8E 00 04 01 02 03 04 00
Card answer : 69-85
Reference Implementation answer : 69-85
Comparison status: PASSED

Command: SELECT
Command APDU: 00 A4 04 00 07 A0 00 00 00 04 10 10
Card answer : 6F 17 84 07 A0 00 00 00 04 10 10 A5 0C 50 0A 4D 61 73 74 65 72 43 61 72 64 90-00
Reference Implementation answer : 6F 17 84 07 A0 00 00 00 04 10 10 A5 0C 50 0A 4D 61 73 74 65 72 43 61 72 64 90-00
Comparison status: PASSED

Command: GET PROCESSING OPTIONS
Command APDU: 80 A8 00 00 02 83 00
Card answer : 77 8A 82 02 00 00 94 04 08 01 01 00 90-00
Reference Implementation answer : 77 8A 82 02 00 00 94 04 08 01 01 00 90-00
Comparison status: PASSED

Command: GET PROCESSING OPTIONS
Command APDU: 80 A8 00 00 02 83 00
Card answer : 69-85
Reference Implementation answer : 69-85
Comparison status: PASSED

Command: GET PROCESSING OPTIONS
Command APDU: 80 A8 00 00 02 83 00
Card answer : 69-85
Reference Implementation answer : 77 8A 82 02 00 00 94 04 08 01 01 00 90-00
Comparison status: FAILED
    
```

FIGURE 4.28 – Mise en évidence par l’implémentation de référence d’une non conformité dans l’application cible

SW1	SW2	SELECTED
‘67’	‘00’	SELECTED
‘69’	‘85’	SELECTED
‘6A’	‘86’	SELECTED
‘90’	‘00’	INITIATED
Other		SELECTED

FIGURE 4.29 – Matrice des états de destination de la commande GET PROCESSING OPTIONS



```
file:///C:/Documents and Settings/valimi/My Documents/Visual Studio 2010/Projects/WinSCard Tools [git]/WSCT Fuzzing/WSCT.Fuzzing.EMV.ConsoleTests/bin/...
Card answer : 77 00 02 02 00 00 94 04 00 01 01 00 90-00
Reference Implementation answer : 77 00 02 02 00 00 94 04 00 01 01 00 90-00
Comparison status: PASSED

Command: SELECT
Command APDU: 00 A4 04 00 07 00 00 00 00 04 10 10
Card answer : 6F 17 04 07 00 00 00 00 04 10 10 05 0C 50 00 4D 61 73 74 65 72 43 61 72 64 90-00
Reference Implementation answer : 6F 17 04 07 00 00 00 00 04 10 10 05 0C 50 00 4D 61 73 74 65 72 43 61 72 64 90-00
Comparison status: PASSED

Command: COMPUTE CRYPTOGRAPHIC CHECKSUM
Command APDU: 80 2A 8E 00 04 01 02 03 04 00
Card answer : 69-85
Reference Implementation answer : 69-85
Comparison status: PASSED

Command: COMPUTE CRYPTOGRAPHIC CHECKSUM
Command APDU: 80 2A 8E 00 04 01 02 03 04 00
Card answer : 69-85
Reference Implementation answer : 69-85
Comparison status: PASSED

Command: COMPUTE CRYPTOGRAPHIC CHECKSUM
Command APDU: 80 2A 8E 00 04 01 02 03 04 00
Card answer : 69-85
Reference Implementation answer : 69-85
Comparison status: PASSED

Iteration #54:

Command: SELECT
Command APDU: 00 A4 04 00 07 00 00 00 00 04 10 10
Card answer : 6F 17 04 07 00 00 00 00 04 10 10 05 0C 50 00 4D 61 73 74 65 72 43 61 72 64 90-00
Reference Implementation answer : 6F 17 04 07 00 00 00 00 04 10 10 05 0C 50 00 4D 61 73 74 65 72 43 61 72 64 90-00
Comparison status: PASSED

Command: READ RECORD
Command APDU: 00 B2 00 47 00
Card answer : 6A-86
Reference Implementation answer : 6A-86
Comparison status: PASSED

Command: GET DATA
Command APDU: 80 CA 00 01 00
Card answer : 6A-88
Reference Implementation answer : 69-85
Comparison status: FAILED
```

FIGURE 4.30 – Acceptation de la GET DATA pendant la phase d'utilisation

phase de personnalisation et avec la valeur '00CF' pour le couple P1/P2. Comme le montre la figure 4.30, l'application cible accepte cette commande pendant la phase d'utilisation, *i.e.* après la phase de personnalisation.

Enfin, l'application cible présente une anomalie sur le traitement de la commande READ RECORD. Comme illustré aux figures 4.31 et 4.32, celle-ci retourne un résultat lorsque l'on essaie de lire l'enregistrement '01' d'un fichier dont l'identifiant est supérieur à '0A', alors que la spécification n'autorise la lecture que de l'enregistrement '01' dans le fichier ayant l'identifiant '01' (seul enregistrement personnalisé pour cette application).

L'utilisation d'une application simple telle que MasterCard Paypass Magstripe nous a permis de fournir quelques éléments de validation de l'approche proposée. Nous avons démontré que l'utilisation de la matrice d'acceptation des commandes comme fil conducteur du fuzzing permettait de détecter des vulnérabilités qui ne l'auraient certainement pas été en suivant uniquement la cinématique d'une transaction du point de vue du terminal.

Une perspective envisagée est l'application de la méthode proposée à une application de paiement EMV complète qui serait développée au sein du laboratoire GREYC ou en utilisant l'implémentation EMV open source *OpenEMV* [174].

```

file:///C:/Documents and Settings/valimi/My Documents/Visual Studio 2010/Projects/WinSCard Tools [git]/WSCT Fuzzing/WSCT.Fuzzing.EMV.ConsoleTests/bin/...
Command: READ RECORD
Command APDU: 00 B2 01 51 00
Card answer : 6A-86
Reference Implementation answer : 6A-86
Comparison status: PASSED

Command: READ RECORD
Command APDU: 00 B2 01 52 00
Card answer : 6A-86
Reference Implementation answer : 6A-86
Comparison status: PASSED

Command: COMPUTE CRYPTOGRAPHIC CHECKSUM
Command APDU: 80 2A 8E 80 04 01 02 03 04 00
Card answer : 69-85
Reference Implementation answer : 69-85
Comparison status: PASSED

Iteration #256:

Command: SELECT
Command APDU: 00 A4 04 00 07 A0 00 00 00 04 10 10
Card answer : 6F 17 84 07 A0 00 00 00 04 10 10 A5 0C 50 00 4D 61 73 74 65 72 43 61 72 64 90-00
Reference Implementation answer : 6F 17 84 07 A0 00 00 00 04 10 10 A5 0C 50 00 4D 61 73 74 65 72 43 61 72 64 90-00
Comparison status: PASSED

Command: GET PROCESSING OPTIONS
Command APDU: 80 A8 00 00 02 83 00
Card answer : 77 00 82 02 00 00 94 04 08 01 01 00 90-00
Reference Implementation answer : 77 00 82 02 00 00 94 04 08 01 01 00 90-00
Comparison status: PASSED

Command: READ RECORD
Command APDU: 00 B2 01 53 00
Card answer : 6A-86
Reference Implementation answer : 6A-86
Comparison status: PASSED

Command: READ RECORD
Command APDU: 00 B2 01 54 00
Card answer : 70 7F 9F 6C 02 00 01 56 3E 42 35 34 31 33 31 32 33 34 35 36 37 38 34 38 30 30 5E 53 55 50 50
4C 49 45 44 2F 4E 4F 54 5E 30 39 30 36 31 30 31 33 33 30 30 30 33 33 33 30 30 30 32 32 32 32 30 30 30 31 31 31 31 30 9F 64 01 03
9F 62 06 00 00 00 30 00 00 9F 63 06 00 00 00 00 E0 E0 9F 65 02 00 0E 9F 66 02 0E 70 9F 60 13 54 13 12 34 56 70 48 00 00 90 61 01 90
00 9F 00 00 00 02 9F 67 01 03 90-00
Reference Implementation answer : 6A-82
Comparison status: FAILED
    
```

FIGURE 4.31 – Acceptation de la commande READ RECORD pour l'enregistrement '01' dans le fichier '0A'

```

file:///C:/Documents and Settings/valimi/My Documents/Visual Studio 2010/Projects/WinSCard Tools [git]/WSCT Fuzzing/WSCT.Fuzzing.EMV.ConsoleTests/bin/...
Command: GET PROCESSING OPTIONS
Command APDU: 80 A8 00 00 02 83 00
Card answer : 77 00 82 02 00 00 94 04 08 01 01 00 90-00
Reference Implementation answer : 77 00 82 02 00 00 94 04 08 01 01 00 90-00
Comparison status: PASSED

Command: GET PROCESSING OPTIONS
Command APDU: 80 A8 00 00 02 83 00
Card answer : 69-85
Reference Implementation answer : 69-85
Comparison status: PASSED

Command: GET PROCESSING OPTIONS
Command APDU: 80 A8 00 00 02 83 00
Card answer : 69-85
Reference Implementation answer : 77 00 82 02 00 00 94 04 08 01 01 00 90-00
Comparison status: FAILED

Command: SELECT
Command APDU: 00 A4 04 00 07 A0 00 00 00 04 10 10
Card answer : 6F 17 84 07 A0 00 00 00 04 10 10 A5 0C 50 00 4D 61 73 74 65 72 43 61 72 64 90-00
Reference Implementation answer : 6F 17 84 07 A0 00 00 00 04 10 10 A5 0C 50 00 4D 61 73 74 65 72 43 61 72 64 90-00
Comparison status: PASSED

Command: COMPUTE CRYPTOGRAPHIC CHECKSUM
Command APDU: 80 2A 8E 80 04 01 02 03 04 00
Card answer : 69-85
Reference Implementation answer : 69-85
Comparison status: PASSED

Iteration #262:

Command: SELECT
Command APDU: 00 A4 04 00 07 A0 00 00 00 04 10 10
Card answer : 6F 17 84 07 A0 00 00 00 04 10 10 A5 0C 50 00 4D 61 73 74 65 72 43 61 72 64 90-00
Reference Implementation answer : 6F 17 84 07 A0 00 00 00 04 10 10 A5 0C 50 00 4D 61 73 74 65 72 43 61 72 64 90-00
Comparison status: PASSED

Command: READ RECORD
Command APDU: 00 B2 01 5C 00
Card answer : 70 7F 9F 6C 02 00 01 56 3E 42 35 34 31 33 31 32 33 34 35 36 37 38 34 38 30 30 5E 53 55 50 50
4C 49 45 44 2F 4E 4F 54 5E 30 39 30 36 31 30 31 33 33 30 30 30 33 33 33 30 30 30 32 32 32 32 30 30 30 31 31 31 31 30 9F 64 01 03
9F 62 06 00 00 00 30 00 00 9F 63 06 00 00 00 00 E0 E0 9F 65 02 00 0E 9F 66 02 0E 70 9F 60 13 54 13 12 34 56 70 48 00 00 90 61 01 90
00 9F 00 00 00 02 9F 67 01 03 90-00
Reference Implementation answer : 6A-82
Comparison status: FAILED
    
```

FIGURE 4.32 – Acceptation de la commande READ RECORD pour l'enregistrement '01' dans le fichier '0B'

4.5.5 Discussion

Le fuzzing par algorithme génétique est une approche intéressante car elle permet un auto-ajustement des données envoyées à l'application afin de découvrir une vulnérabilité. Cependant, l'inconvénient de son utilisation dans ce contexte réside dans la grande difficulté à évaluer la qualité des individus représentant les données des commandes envoyées et à sélectionner les plus prometteurs.

L'approche par blocs et par implémentation de référence, en utilisant la matrice d'acceptation des commandes, permet de réaliser un test en profondeur des applications d'un point de vue fonctionnel et de réaliser les mêmes tests sécuritaires que dans l'approche proposée par Lancia. L'implémentation de référence permet à un instant t , pour une transaction donnée, de connaître le niveau de conformité de l'application par rapport à la spécification et, dans le cas contraire, de connaître exactement l'écart d'implémentation. Notre contribution est ici très appliquée au domaine du paiement EMV mais est généralisable. L'effort d'adaptation à un autre type d'application consiste à la définition de la matrice d'acceptation des commandes (état/commande) de l'application ciblée et des du format des données de ces commandes (contenu, format, valeurs limites).

Pour des raisons de temps, notre framework de fuzzing n'a pas été testé sur des implémentations EMV complètes mais sur une application EMV partielle et une application *online*. Le but de ces travaux était plus de démontrer la faisabilité d'un tel framework de fuzzing que de réaliser une implémentation complète. Ces travaux seront poursuivis dans le cadre d'une prochaine thèse réalisée au sein de l'équipe *Monétique et Biométrie* du laboratoire GREYC. Une perspective d'évolution déjà identifiée est l'utilisation combinée des algorithmes génétiques et de l'implémentation de référence. L'implémentation de référence permettrait ainsi d'obtenir une évaluation très fine pour quantifier la qualité des individus en utilisant ses indicateurs internes. Une autre perspective identifiée est l'amélioration de l'implémentation de référence par l'utilisation de langages de propriétés temporelles [175, 176].

4.6 Conclusion

Nous avons détaillé dans ce chapitre nos contributions au logiciel WinsCard Tools. Développé à l'ENSICAEN à des fins pédagogiques puis de recherche, nous l'avons étendu à l'aide de modules complémentaires lui permettant aujourd'hui d'être considéré comme une plateforme logicielle de test et d'analyse des transactions par carte à puce à contact et sans contact. WSCT permet de reproduire facilement des attaques de l'état de l'art telles que les attaques *man-in-the-middle* sur la vérification

du code PIN – attaque de Cambridge – ou sur les données de transaction – attaque sur une transaction SDA. Cette dernière attaque a été rendue possible en connectant une sonde sans contact et en développant une couche d'adaptation. De par l'architecture modulaire de WSCT, ces développements et leur mise en œuvre ont été réalisés de manière rapide et efficace. Cette architecture en couches, proche du modèle des fuzzers, et toutes les briques de base présentes dans WSCT, ont facilité la conception et le développement d'un framework de fuzzing appliqué à la carte à puce et plus particulièrement aux applications de paiement EMV.

Une première approche utilisant les algorithmes génétiques pour la génération et l'évaluation des données de fuzzing a été mise en œuvre. Bien que donnant des résultats, elle a mis en évidence la difficulté à évaluer les données et à interpréter les résultats dans le contexte global de la campagne de fuzzing. Nous nous sommes alors tourné vers une seconde approche inspirée des travaux de Julien Lancia à SERMA Technologies et basée sur une implémentation de référence [157]. Nous avons proposé une amélioration de cette approche permettant de réaliser un test en profondeur des applications en utilisant la matrice d'acceptation des commandes et la machine d'état propres à chaque application.

Cette seconde approche n'a pu être mise en œuvre sur une application de paiement EMV complète mais sur une application *online* dont la complexité est réduite. Notre objectif était tout d'abord de proposer une méthodologie et d'implémenter une preuve de concept afin de valider cette méthodologie. Ces travaux seront repris et approfondis au cours d'une prochaine thèse réalisée au sein de l'équipe *Monétique et Biométrie*. Des perspectives et des axes de recherche ont déjà été identifiés comme l'utilisation de langages de propriétés temporelles pour améliorer l'implémentation de référence et l'utilisation combinée des algorithmes génétiques et de l'implémentation de référence pour l'évaluation des chromosomes.

Conclusions et perspectives

Conclusion

Nous avons étudié dans ce mémoire les services mobiles sans contact au long de leur cycle de vie : développement, déploiement et utilisation. Nous avons proposé des contributions dans chacune de ces phases.

Pour la phase de développement, nous proposons un framework de fuzzing permettant aux développeurs d'applets de réaliser les tests fonctionnels en profondeur et à moindre coût. Il tire pleinement parti de l'outil WinSCard Tools développé au laboratoire GREYC. Nous proposons l'utilisation d'une implémentation de référence de l'application à tester et d'une matrice d'acceptation des commandes. Des paramètres permettent d'affiner la granularité des tests de la machine d'état.

Pour la phase de déploiement, nous proposons un système de génération automatique de scripts de déploiement. Ce système est basé sur une modélisation à deux dimensions par les ontologies OWL de la spécification GlobalPlatform. Pour optimiser le processus de déploiement, nous proposons que le Secure Element lui-même stocke les données qui le caractérisent et que le TSM les récupère directement avant de les injecter dans le modèle. Nous en proposons une implémentation et une validation à la fois sur des bancs d'évaluation et sur des équipements mobiles réels.

La récupération des données de configuration du Secure Element et l'ouverture d'un canal de communication sécurisé par le TSM sont soumis à la condition de connaître les clés qui protègent le Secure Element. Nous proposons un schéma de distribution de clés du Secure Element à la demande. Ce schéma met en œuvre un tiers de confiance chargé de la gestion des clés pour un émetteur. Après s'être authentifiés, les TSMs lui demandent les clés de leur domaine de sécurité puis effectuent une rotation de clés afin de « prendre possession des lieux ». Nous avons soumis ce schéma à l'organisation de standardisation GlobalPlatform et suivons les différentes étapes

amenant à l'ajout des différents éléments nécessaires dans la prochaine version des spécifications.

Notre contribution se place également de côté de l'acceptation des services mobiles. Nous proposons trois architectures permettant à un téléphone NFC d'accepter des transactions de paiement sans contact. Les trois composants essentiels sont l'application de paiement responsable de mener la transaction de paiement avec une carte sans contact ou mobile, un *Secure Access Module* stockant de manière sécurisée des clés, et un élément de sécurité capable de réaliser des opérations cryptographiques. Une première solution consiste à mettre tous les composants dans un module externe au mobile, une seconde consiste en une solution hybride avec l'application de paiement dans le processeur d'application et la cryptographie dans le SE. La dernière solution consiste à mettre tous les composants dans le SE. Malgré l'effort d'intégration à fournir par les fabricants de mobiles, elles offrent un niveau de sécurité satisfaisant. Cependant, la solution hybride semble se démarquer notamment grâce au concept de *Trusted Execution Environment*.

Pour la phase d'utilisation du service mobile, nous proposons une plateforme d'analyse de la sécurité des transactions. Nous agrémentons l'outil existant WinSCard Tools de nombreux modules qui lui permettent de reproduire des attaques de l'état de l'art telle que l'attaque de Cambridge. Un autre module lui permet également de réaliser des attaques *man-in-the-middle* sur des transactions sans contact de type SDA. Enfin, nous proposons un framework de fuzzing pour réaliser des attaques en boîte noire sur des applications déployées. Deux méthodes sont étudiées : l'une basée sur les algorithmes génétiques et l'autre basée sur une implémentation de référence. Cette dernière, améliore l'existant en ne prenant pas en compte l'application uniquement dans le contexte de la transaction de paiement mais en prenant la matrice d'acceptation des commandes des applications comme fil conducteur du fuzzing.

Perspectives de la thèse

Le framework WinSCard Tools permet de s'interfacer avec tous types de cartes conformes aux standards ISO 7816 mais est aujourd'hui très orientée applications de cartes de paiement du fait de l'histoire et des activités de l'équipe. WinSCard Tools va évoluer pour intégrer des bibliothèques permettant de réaliser la lecture, l'interprétation des données et du fuzzing d'autres types d'applications telles que le transport, l'identité ou le porte-monnaie électronique.

L'architecture de terminal de paiement mobile a fait l'objet d'améliorations afin de se conformer aux exigences sécuritaires du consortium PCI en termes d'acceptation de

paiements mobiles de proximité. Ces améliorations, propriété industrielle d'INSIDE Secure, sont confidentielles et ne peuvent être exposées dans ce mémoire. Cependant, elles feront l'objet de publications et de présentations après le dépôt et l'acceptation des différentes demandes de brevets applicables. A l'issue de cette thèse, je serai employé par INSIDE Secure afin d'être le chef de produit de cette solution. Mon rôle consistera à rédiger les spécifications techniques et encadrer les phases de développement et de validation.

Nous avons vu que l'authentification du porteur lors d'un paiement mobile de proximité, repose notamment sur la saisie d'un code personnel sur le clavier de l'équipement mobile. Ce dernier est ensuite présenté devant le terminal de paiement afin de procéder à la transaction. Mais ce mécanisme n'est pas satisfaisant car le porteur n'est pas réellement authentifié. En fait, la seule chose que l'on peut affirmer est qu'un code a été saisi sur l'équipement mobile, ce qui a eu effet de débloquent la fonctionnalité de paiement. Pour obtenir une authentification réelle du porteur, faut « aller jusqu'au porteur », *i.e.* lui demander une information qui le caractérise et l'authentifie réellement. Nous avons proposé dans [177], une méthode d'authentification de type *match-on-card* du porteur par sa dynamique de signature. Nous envisageons d'utiliser cette solution à la fois sur le terminal mobile du marchand pour dématérialiser la signature sur la facturette en soumettant la signature capturée sur l'écran directement à la carte du porteur; et sur le mobile du porteur pour permettre de s'authentifier auprès d'une multitude services. Dans ce dernier cas, la signature capturée serait envoyée au Secure Element pour vérification.

L'implémentation de référence de l'application à tester avec le framework de fuzzing peut être améliorée en y ajoutant des notions de vérification formelle. En effet, grâce au système d'observateurs mis en œuvre dans WinSCard Tools, l'implémentation de référence peut se voir notifier en temps réel de la réalisation ou non de certaines conditions et de la mise à jour d'états internes. Il est également envisagé de coupler l'implémentation de référence aux algorithmes génétiques. En effet, l'implémentation de référence est la plus pertinente pour calculer un score en fonction de l'état de ses indicateurs internes et de ses propriétés.

Bibliographie

- [1] Fédération des Entreprises de Vente à Distance (FEVAD). Chiffres clés vente à distance e-commerce - édition 2011. http://www.fevad.com/uploads/files/Etudes/fevad2011_chiffres.pdf, 2011. [cité p. 2]
- [2] Autorité de Régulation des Communications Et Postes (ARCEP). Observatoire trimestriel des marchés des communications électroniques (services mobiles) en france - 4ème trimestre 2010. <http://www.arcep.fr/index.php?id=10743&L=0>, 2011. [cité p. 2]
- [3] Groupement Carte Bancaire. Rapport d'activité carte bancaire 2010. http://www.cartes-bancaires.com/IMG/pdf/Rapport_d_Activite_CB_2010.pdf, 2011. [cité p. 2]
- [4] francemobiles.com. Le téléphone mobile, un outil indispensable! http://www.francemobiles.com/actualites/id/200909081252311197/le_telephone_mobile_un_outil_indispensable_.html, 2009. [cité p. 2]
- [5] MaxiSciences. Le mobile, outil indispensable pour les français? http://www.maxisciences.com/t%E9I%E9phone-mobile/le-mobile-outil-indispensable-pour-les-francais_art9851.html, 2010. [cité p. 2]
- [6] Wikipédia. 3g. <http://fr.wikipedia.org/wiki/3G>, 2011. [cité p. 2]
- [7] Damien Mathieu. Déployer une application android. <http://dmathieu.com/fr/android/deployer-une-application-android>, 2010. [cité p. 3]
- [8] Wikipédia. Jailbreak d'ios. http://fr.wikipedia.org/wiki/Jailbreak_d'iOS, 2011. [cité p. 3]
- [9] Oracle. Understanding midp 2.0's architecture. <http://developers.sun.com/mobility/midp/articles/permissions/>, 2003. [cité p. 4]
- [10] GlobalPlatform. Globalplatform. <http://www.globalplatform.org/>. [cité p. 4]

- [11] Marie Reveilhac and Marc Pasquet. Promising secure element alternatives in nfc architecture. In *Proceedings of NFC Congress 2009*, pages 0–6, Hagenberg, Austria, 2008. IEEE Computer Society. [cité p. 5, 30]
- [12] Gerald Madlmayr, Oliver Dillinger, Josef Langer, and Christoph Schaffer. The benefit of using sim application toolkit in the context of near fieldcommunication applications. *Sixth International Conference on the Management of Mobile Business*, 2007. [cité p. 5, 22]
- [13] Autorité de régulation des communications électroniques et des postes. Observatoire trimestriel des marchés des communications électroniques (services mobiles) en france - 3ème trimestre 2010 - résultats provisoires - publication le 4 novembre 2010. <http://www.arcep.fr/index.php?id=35>, 2010. [cité p. 7]
- [14] Gerald Madlmayr and Josef Langer. Managing an nfc ecosystem. *7th International Conference on Mobile Business*, 2008. [cité p. 7, 17, 123]
- [15] GSM Association. *White paper on NFC services*, 2007. [cité p. 7, 31]
- [16] Encyclopédie Universalis. Télécommunications - la communication sans fil, le téléphone mobile, « couteau suisse » du sans-fil. <http://www.universalis.fr/encyclopedie/telecommunications-la-communication-sans-fil/4-le-telephone-mobile-couteau-suisse-du-sans-fil/>, 2010. [cité p. 10]
- [17] Pôle Transaction Electroniques Sécurisées. Les transactions électroniques sécurisées : un enjeu clé de la société de l’information de demain. Technical report, Pôle TES, 2006. [cité p. 10]
- [18] Stéphane Bresson. Mémoire sur les transactions électroniques sécurisées et la monétique. Technical report, Centres des Technologies Nouvelles, 2004. [cité p. 10]
- [19] Pôle Transaction Electroniques Sécurisées. Les transactions électroniques sécurisées : un enjeu clé de la société de l’information de demain. Technical report, Pôle Transaction Electroniques Sécurisées, 2006. [cité p. 10, 193]
- [20] de lévaluation des politiques publiques et du développement de léconomie numérique Premier Ministre, Secrétariat d’Etat chargé de la prospective. *France numérique 2012, plan de développement de léconomie numérique*, 2008. [cité p. 11]
- [21] Autorité de régulation des communications électroniques et des postes. Contactless mobile services. Technical report, ARCEP, 2010. [cité p. 11]
- [22] Wikipédia. Mobile payment. http://en.wikipedia.org/wiki/Mobile_payment, 2011. [cité p. 12]
- [23] Mahil Carr. Mobile payment systems and services : an introduction. Technical report, IDRBT. [cité p. 12]

- [24] Cédric Descoutures. M-payment - les différents types de m-payment, leurs spécificités et perspectives d'avenir. <http://www.slideshare.net/cdescoutures/les-differents-types-de-mpayment-support-ppt>, 2009. [cité p. 12]
- [25] E. van der Linde and G.P. Hancke. An investigation of bluetooth mergence with ultra wideband. *Ad Hoc Networks*, 9(5) :852 – 863, 2011. [cité p. 13]
- [26] Naveen Erasala and David C. Yen. Bluetooth technology : a strategic analysis of its role in global 3g wireless communication era. *Computer Standards & Interfaces*, 24(3) :193 – 206, 2002. [cité p. 13]
- [27] Wikipédia. Bluetooth. <http://fr.wikipedia.org/wiki/Bluetooth>, 2011. [cité p. 13]
- [28] Sojen Pradhan, Elaine Lawrence, and Agnieszka Zmijewska. Bluetooth as an enabling technology in mobile transactions. *Information Technology : Coding and Computing, International Conference on*, 2 :53–58, 2005. [cité p. 14]
- [29] Wendell Odom. *CCNA Official Exam Certification Library (CCNA Exam 640-802)*. Cisco Press, 2007. [cité p. 14]
- [30] John Bellardo and Stefan Savage. 802.11 denial-of-service attacks : real vulnerabilities and practical solutions. In *Proceedings of the 12th conference on USENIX Security Symposium - Volume 12*, pages 2–2, Berkeley, CA, USA, 2003. USENIX Association. [cité p. 15]
- [31] Chris Karlof and David Wagner. Secure routing in wireless sensor networks : attacks and countermeasures. *Ad Hoc Networks*, 1(2-3) :293 – 315, 2003. <ce :title>Sensor Network Protocols and Applications</ce :title>. [cité p. 15]
- [32] Adrian Perrig, John Stankovic, and David Wagner. Security in wireless sensor networks. *Commun. ACM*, 47 :53–57, June 2004. [cité p. 15]
- [33] Halil Ibrahim Bulbul, Ihsan Batmaz, and Mesut Ozel. Wireless network security : comparison of wep (wired equivalent privacy) mechanism, wpa (wi-fi protected access) and rsn (robust security network) security protocols. In *Proceedings of the 1st international conference on Forensic applications and techniques in telecommunications, information, and multimedia and workshop, e-Forensics '08*, pages 9 :1–9 :6, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). [cité p. 15]
- [34] Agnieszka Zmijewska. Evaluating wireless technologies in mobile payments : A customer centric approach. *Mobile Business, International Conference on*, 0 :354–362, 2005. [cité p. 15, 21, 51]
- [35] Dr K. Finkenzerler. *RFID Handbook : Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication, Third Edition*. John Wiley and Sons, 2010. [cité p. 15, 123]

- [36] ISO/IEC, <http://www.iso.org>. *ISO/IEC 14443 : Identification cards – Contactless integrated circuit(s)cards – Proximity cards*. [cité p. 16]
- [37] ISO/IEC, <http://www.iso.org>. *ISO/IEC 15693 : Identification cards – Contactless integrated circuit(s)cards – Vicinity cards*. [cité p. 17]
- [38] Vincent Alimi and Marc Pasquet. Post-distribution provisioning and personalization of a payment application on a UICC-based Secure Element. In *1st International Workshop on Sensor Security*, 2009. [cité p. 18]
- [39] Gerald Madlmayr, Josef Langer, Christian Kantner, and Josef Scharinger. Nfc devices : Security and privacy. *Availability, Reliability and Security, International Conference on*, 0 :642–647, 2008. [cité p. 18, 20]
- [40] Yi Lu, Willi Meier, and Serge Vaudenay. The conditional correlation attack : A practical attack on bluetooth encryption. In Victor Shoup, editor, *Advances in Cryptology CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 97–117. Springer Berlin / Heidelberg, 2005. [cité p. 19]
- [41] Jonathan Wheeler John Jersin. Security analysis of bluetooth v2.1 + edr pairing authentication protocol. Technical report, Stanford University, 2008. [cité p. 19]
- [42] Y.C. Justin Wan William Arbaugh, Narendar Shankar and Kan Zhang. Your 80211 wireless network has no clothes. *Wireless Communications, IEEE*, 9(6) :44 – 51, dec. 2002. [cité p. 20]
- [43] Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting mobile communications : the insecurity of 802.11. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, MobiCom '01, pages 180–189, New York, NY, USA, 2001. ACM. [cité p. 20]
- [44] Adam Stubblefield, John Ioannidis, John Ioannidis, Aviel D. Rubin, and Aviel D. Rubin. Using the fluhrer, mantin, and shamir attack to break wep, 2001. [cité p. 20]
- [45] Dominik and Alain Girardet. Wepattack. <http://sourceforge.net/projects/wepattack/>, 2011. [cité p. 20]
- [46] Ernst Haselsteiner and Klemens Breitfuss. *Security in Near Field Communication (NFC)*. 2006. [cité p. 20]
- [47] ECMA. Ecma-385 – nfc-sec : Nfcip-1 security services and protocol. Technical report, European Computer Manufacturers Association, 2008. [cité p. 20]
- [48] Kristen Moyer. Five reasons bluetooth mobile payments won't happen. http://blogs.gartner.com/kristin_moyer/2009/01/15/five-reasons-bluetooth-mobile-payments-wont-happen/, 2009. [cité p. 20]

- [49] EMVCo. Emv mobile contactless payment technical issues and position paper. Technical report, EMVCo, 2007. [cité p. 23]
- [50] Oracle. Java card technology. <http://www.oracle.com/technetwork/java/javacard/-overview/index.html>, 2010. [cité p. 23]
- [51] GlobalPlatform. *GlobalPlatform Card Specification Version 2.2*, 2006. [cité p. 23, 26, 60]
- [52] Tanenbaum and Andrew S. *Modern Operating Systems*. Prentice Hall Press, Upper Saddle River, NJ, USA, 2007. [cité p. 23]
- [53] W. Rankl and W. Effing. *Smart Card Handbook*. John Wiley & Sons, Inc., New York, NY, USA, 2003. [cité p. 23, 104]
- [54] EMVCo. EMV integrated circuit card specifications for payment systems. Technical report, EMVCo, 2008. [cité p. 25, 50, 116]
- [55] Watt, David A, Brown, and Deryck F. *Programming language processors in Java : compilers and interpreters*. Prentice Hall, 2000. [cité p. 25]
- [56] GlobalPlatform. *GlobalPlatform Card v2.2 Amendment A - Confidential Card Content Management*, 2007. [cité p. 26]
- [57] GlobalPlatform. *The Trusted Execution Environment – Delivering Enhanced Security at a Lower Cost to the Mobile Market*, 2011. [cité p. 26, 29, 53, 84]
- [58] GlobalPlatform. *Messaging Specification for Management of Mobile-NFC Services*, 2011. [cité p. 26, 45]
- [59] GlobalPlatform. *Key Management Requirements Systems Functional Requirements Specification*. http://www.globalplatform.org/specifications/systems/KMS_Functional_Reqs_v1_0.pdf. [cité p. 26]
- [60] Marten Van Dijk, Luis F. G. Sarmenta, Jonathan Rhodes, and Srinivas Devadas. Securing shared untrusted storage by using tpm 1.2 without requiring a trusted os. Technical report, 2007. [cité p. 29]
- [61] Luis F. G. Sarmenta, Marten van Dijk, Charles W. O'Donnell, Jonathan Rhodes, and Srinivas Devadas. Virtual monotonic counters and count-limited objects using a tpm without a trusted os. In *STC*, pages 27–42, 2006. [cité p. 29]
- [62] OMTP. Advanced trusted environment : Omtpt r1. Technical report, Open Mobile Terminal Platform, 2009. [cité p. 29]
- [63] GlobalPlatform. *GlobalPlatform Device Technology – TEE Client API Specification*. GlobalPlatform. [cité p. 29]

- [64] Keith E. Mayes, Konstantinos Markantonakis, and Keith Mayes. An introduction to smart cards. In *Smart Cards, Tokens, Security and Applications*, pages 1–25. Springer US, 2008. [cité p. 30]
- [65] GlobalPlatform. *GlobalPlatform’s Proposition for NFC Mobile : Secure Element Management and Messaging*, 2009. [cité p. 31]
- [66] Lina Alchaal and Vincent Roca. Managing and securing web services with vpns. In *Proceedings of the IEEE International Conference on Web Services (ICWS04)*, 2004. [cité p. 42]
- [67] GlobalPlatform. *GlobalPlatform Card v2.2 Amendment B - Remote Application Management over HTTP*, 2007. [cité p. 42, 43, 87]
- [68] GlobalPlatform. About Mobile Task Force. <http://www.globalplatform.org/aboutustaskforcesmobile.asp>. [cité p. 44]
- [69] Payment Card Industry Security Standards Council. Payment card industry data security standard, October 2010. [cité p. 46]
- [70] Elitt. Payment card industry(pci). <http://www.elitt.com/fr/payment-card-industry-pci-t146.html>. [cité p. 47]
- [71] Payment Card Industry Security Standards Council. Pci security standards council update on pa-dss and mobile payment acceptance applications. https://www.pcisecuritystandards.org/documents/statement_110624_pcissc.pdf, June 2011. [cité p. 49]
- [72] Tomi Dahlberg, Niina Mallat, Jan Ondrus, and Agnieszka Zmijewska. Past, present and future of mobile payments research : A literature review. *Electronic Commerce Research and Applications*, 7(2) :165–181, 2008. [cité p. 51]
- [73] Jan Ondrus and Yves Pigneur. An assessment of nfc for future mobile payment systems. In *Proceedings of the International Conference on the Management of Mobile Business*, Washington, DC, USA, 2008. IEEE Computer Society. [cité p. 51]
- [74] Jiajun Jim Chen and Carl Adams. Short-range wireless technologies with mobile payments systems. In *Proceedings of the 6th international conference on Electronic commerce, ICEC '04*, pages 649–656, New York, NY, USA, 2004. ACM. [cité p. 51]
- [75] Florian Resatsch, Stephan Karpischek, Uwe Sandner, and Stephan Hamacher. Mobile sales assistant : Nfc for retailers. In *Proceedings of the 9th international conference on Human computer interaction with mobile devices and services, MobileHCI '07*, pages 313–316, New York, NY, USA, 2007. ACM. [cité p. 51]
- [76] Common criteria : The common criteria portal. <http://www.commoncriteriaportal.org/>. [cité p. 57]

- [77] GlobalPlatform. Globalplatform defines new certification model for mobile secure elements. <http://www.globalplatform.org/mediapressview.asp?id=868>, May 2011. [cité p. 57]
- [78] Paul Beynon-Davies. *Database Systems 3rd Edition*. Palgrave Macmillan, 2004. [cité p. 61]
- [79] ANSI/X3/SPARC Study Group on Data Base Management Systems. Interim report. Technical report, American National Standards Institute, 1975. [cité p. 62]
- [80] Janis A. Bubenko Jr. Information system methodologies - a research view. In *Information Systems Design Methodologies : Improving the Practice*, pages 289–312, 1986. [cité p. 63]
- [81] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13 :377–387, June 1970. [cité p. 63]
- [82] Laurent Audibert. Qu'est-ce qu'une base de données? <http://laurent-audibert.developpez.com/Cours-BD/html/Cours-BD005.html>. [cité p. 63]
- [83] ISO/IEC. *ISO/IEC 9075 : Technologies de l'information - Langages de base de données - SQL*. [cité p. 63]
- [84] Le modèle relationnel. <http://www.commentcamarche.net/contents/relation/-relintro.php3>, 2008. [cité p. 63]
- [85] Laurent Audibert. Base de données et langage sql. <http://laurent-audibert.developpez.com/Cours-BD/html/Cours-BD014.php>. [cité p. 63]
- [86] Sharon Allen and Evan Terry. Introducing object-oriented data modeling. In *Beginning Relational Data Modeling*, pages 107–129. Apress, 2005. 10.1007/978-1-4302-0015-4_5. [cité p. 65]
- [87] Nicols Holzschuch. Modèles hiérarchiques. <http://artis.imag.fr/Nicolas.Holzschuch/cours/class2.ppt>. [cité p. 65]
- [88] Wikipedia. Diagramme de classes. http://fr.wikipedia.org/wiki/Diagramme_de_classes. [cité p. 66]
- [89] Grady Booch. *Object oriented design with applications*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1991. [cité p. 66]
- [90] M. Loomis, A. Shah, and J. Rumbaugh. An object modeling technique for conceptual design. In Jean Bézivin, Jean-Marie Hullot, Pierre Cointe, and Henry Lieberman, editors, *ECOOP 87 European Conference on Object-Oriented Programming*, volume 276 of *Lecture Notes in Computer Science*, pages 192–202. Springer Berlin / Heidelberg, 1987. [cité p. 66]

- [91] I Jacobson, M Christerson, P Jonsson, and G Overgaard. *Object-Oriented Software Engineering : A Use Case Driven Approach*, volume 2640. Addison-Wesley, 1992. [cité p. 66]
- [92] Pierre-Alain Muller. Modélisation objet avec uml. <http://ftp.sliim-projects.eu/docs/Dev/Methodologie/UML/coursUML.pdf>. [cité p. 66]
- [93] D. Akehurst and B. Bordbar. On querying uml data models with ocl. In Martin Gogolla and Cris Kobryn, editors, *UML 2001 The Unified Modeling Language. Modeling Languages, Concepts, and Tools*, volume 2185 of *Lecture Notes in Computer Science*, pages 91–103. Springer Berlin / Heidelberg, 2001. [cité p. 67]
- [94] Anna Queralt and Ernest Teniente. Reasoning on uml class diagrams with ocl constraints. In David Embley, Antoni Olivé, and Sudha Ram, editors, *Conceptual Modeling - ER 2006*, volume 4215 of *Lecture Notes in Computer Science*, pages 497–512. Springer Berlin / Heidelberg, 2006. [cité p. 67]
- [95] John L. Schnase, John J. Leggett, David L. Hicks, and Ron L. Szabo. Semantic data modeling of hypermedia associations. *ACM Transactions on Information Systems*, 11(1) :27–50, January 1993. [cité p. 67]
- [96] Joan Peckham and Fred Maryanski. Semantic data models. *ACM Comput. Surv.*, 20(3) :153–189, September 1988. [cité p. 67]
- [97] Peter P. S. Chen. The entity-relationship model : toward a unified view of data. In *Proceedings of the 1st International Conference on Very Large Data Bases, VLDB '75*, pages 173–173, New York, NY, USA, 1975. ACM. [cité p. 67]
- [98] Viral Parekh, Jin-Ping (Jack) Gwo, and Tim Finin. Ontology based Semantic Metadata for GeoScience Data. In *Proceedings of the International Conference on Information and Knowledge Engineering*, pages 485–490. The International Multi-Conference in Computer Science and Computer Engineering, June 2004. [cité p. 67]
- [99] Jan Jürjens. Modelling audit security for smart-card payment schemes with uml-sec. In *Proceedings of the 16th international conference on Information security : Trusted information : the new decade challenge, Sec '01*, pages 93–107, Norwell, MA, USA, 2001. Kluwer Academic Publishers. [cité p. 69]
- [100] Dominik Haneberg, Holger Grandy, Wolfgang Reif, and Gerhard Schellhorn. Verifying smart card applications : an asm approach. In *Proceedings of the 6th international conference on Integrated formal methods, IFM'07*, pages 313–332, Berlin, Heidelberg, 2007. Springer-Verlag. [cité p. 69]
- [101] Bart Jacobs, Claude Marché, and Nicole Rauch. Formal verification of a commercial smart card applet with multiple tools. In Charles Rattray, Savitri Maharaj, and Carron Shankland, editors, *Algebraic Methodology and Software Technology*, volume

- 3116 of *Lecture Notes in Computer Science*, pages 9–12. Springer Berlin / Heidelberg, 2004. [cité p. 69]
- [102] Fabrice Bouquet, Bruno Legeard, Fabien Peureux, and Eric Torreborre. Mastering test generation from smart card software formal models. In Gilles Barthe, Lilian Burdy, Marieke Huisman, Jean-Louis Lanet, and Traian Muntean, editors, *Construction and Analysis of Safe, Secure, and Interoperable Smart Devices*, volume 3362 of *Lecture Notes in Computer Science*, pages 70–85. Springer Berlin / Heidelberg, 2005. [cité p. 69]
- [103] Santiago Zanella Béguelin. Formalisation and verification of the globalplatform card specification using the b method. In Gilles Barthe, Benjamin Grégoire, Marieke Huisman, and Jean-Louis Lanet, editors, *CASSIS*, volume 3956 of *Lecture Notes in Computer Science*, pages 155–173. Springer, 2005. [cité p. 69]
- [104] GlobalPlatform. *GlobalPlatform Systems Profiles Specification - An XML Representation*. GlobalPlatform, September 2003. [cité p. 69]
- [105] GlobalPlatform. *GlobalPlatform Systems Scripting Language Specification - An ECMAScript Representation*. GlobalPlatform, September 2003. [cité p. 69]
- [106] Eric Boniface. Modèle relationnel. http://nfe113.2bl.fr/documents/2_Modele_relationnel.pdf, 2010. [cité p. 70]
- [107] J.J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena : implementing the semantic web recommendations. pages 74–83, 2004. [cité p. 71]
- [108] H. Chen, T. Finin, and A. Joshi. The soupa ontology for pervasive computing. *Ontologies for agents : Theory and experiences*, pages 233–258, 2005. [cité p. 71]
- [109] G. Tonti, J. Bradshaw, R. Jeffers, R. Montanari, N. Suri, and A. Uszok. Semantic web languages for policy representation and reasoning : A comparison of kaos, rei, and ponder. *The Semantic Web-ISWC 2003*, pages 419–437, 2003. [cité p. 71]
- [110] A. Uszok, J.M. Bradshaw, M. Johnson, R. Jeffers, A. Tate, J. Dalton, and S. Aitken. Kaos policy management for semantic web services. *IEEE Intelligent Systems*, pages 32–41, 2004. [cité p. 71]
- [111] A. Uszok, J.M. Bradshaw, J. Lott, M. Breedy, L. Bunch, P. Feltovich, M. Johnson, and H. Jung. New developments in ontology-based policy management : Increasing the practicality and comprehensiveness of kaos. pages 145–152, 2008. [cité p. 71]
- [112] Jenarules. <http://hydrogen.informatik.tucottbus.de/wiki/index.php/JenaRules>, 2008. [cité p. 72]
- [113] Mircea Diaconescu. Jena rules examples. http://hydrogen.informatik.tucottbus.de/wiki/index.php/Jena_Rules_Examples, 2008. [cité p. 72]

- [114] E. Sirin, B. Parsia, B.C. Grau, A. Kalyanpur, and Y. Katz. Pellet : A practical owl-dl reasoner. *Web Semantics : science, services and agents on the World Wide Web*, 5(2) :51–53, 2007. [cité p. 72]
- [115] Charles L. Forgy. Rete : A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19(1) :17–37, 1982. [cité p. 73]
- [116] S. Harris and A. Seaborne. Sparql 1.1 query language. *W3C Working Draft*, 14, 2010. [cité p. 73]
- [117] What is protégé-owl? Available via the World Wide Web at <http://protege.stanford.edu/overview/protege-owl.html>. [cité p. 74]
- [118] W3C. Rdf vocabulary description language 1.0 : Rdf schema. Technical report, W3C. [cité p. 74]
- [119] Mizoguch Lab. and Ltd ENEGATE Co. Hozo - ontology editor. http://www.ei.sanken.osaka-u.ac.jp/hozo/eng/index_en.php. [cité p. 74]
- [120] Ontoprise GmbH. Ontoprise : Ontostudio. <http://www.ontoprise.de/en/products/-ontostudio/>. [cité p. 74]
- [121] Swoop : Semantic web ontology editor. <http://code.google.com/p/swoop/>. [cité p. 74]
- [122] The Apache Software Foundation. Apache jena. <http://incubator.apache.org/jena/>. [cité p. 74]
- [123] W3C. Sparql query language for rdf. Technical report, W3C. [cité p. 74]
- [124] Jeroen Frijters. IKVM .NET. <http://www.ikvm.net/>, 2011. [cité p. 74]
- [125] Visa International. *Visa GlobalPlatform 2.1.1 Secure Element Specifications*, 2007. [cité p. 79]
- [126] Vincent Alimi. An ontology-based framework to model a globalplatform secure element. In *Near Field Communication (NFC), 2012 4th International Workshop on*, pages 25–30. IEEE, 2012. [cité p. 93]
- [127] Gary McGraw Greg Hoglund. *Exploiting Software : How to Break Code*. Add, 2004. [cité p. 103]
- [128] Cigital. Cigital : Coding rules overview. <https://buildsecurityin.us-cert.gov/bsi-rules/33-BSI.html>. [cité p. 103]
- [129] Hex-Rays. Ida : About. <http://www.hex-rays.com/products/ida/index.shtml>. [cité p. 103]
- [130] Michael Tunstall. Smart card security. In Keith Mayes and Konstantinos Markantonakis, editors, *Smart Cards, Tokens, Security and Applications*, pages 195–228. Springer, 2008. [cité p. 104]

- [131] Vincent Alimi. Etat de l'art de la sécurité de la carte à puce. Technical report, Laboratoire GREYC, 2009. [cité p. 105]
- [132] John Horton. Conway. *On numbers and games / J. H. Conway*. Academic Press, London ; New York :, 1976. [cité p. 105]
- [133] Yvo Desmedt, Claude Goutier, and Samy Bengio. Special uses and sbuses of the fiat-shamir passport protocol. In *A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*, CRYPTO '87, pages 21–39, London, UK, UK, 1988. Springer-Verlag. [cité p. 105]
- [134] Y.C. Hu, A. Perrig, and D.B. Johnson. Wormhole attacks in wireless networks. *Selected Areas in Communications, IEEE Journal on*, 24(2) :370–380, 2006. [cité p. 105]
- [135] Lishoy Francis, Gerhard Hancke, Keith Mayes, and Konstantinos Markantonakis. Practical nfc peer-to-peer relay attack using mobile phones. In *Proceedings of the 6th international conference on Radio frequency identification : security and privacy issues*, RFIDSec'10, pages 35–49, Berlin, Heidelberg, 2010. Springer-Verlag. [cité p. 105]
- [136] Steven Murdoch, Saar Drimer, Ross Anderson, and Mike Bond. Chip and PIN is broken. In David Evans and Giovanni Vigna, editors, *SSP 2010, 31st IEEE Symposium on Security & Privacy*, Piscataway, NJ, USA, May 2010. IEEE Computer Society Technical Committee on Security and Privacy/The International Association for Cryptologic Research, IEEE Computer Society. [cité p. 105, 106, 194]
- [137] Saar Drimer and Steven J. Murdoch. Keep your enemies close : distance bounding against smartcard relay attacks. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, SS'07, pages 7 :1–7 :16, Berkeley, CA, USA, 2007. USENIX Association. [cité p. 107]
- [138] *Gem_PCSC Tool*. <http://support.gemalto.com/?id=199>. vu en novembre 2011. [cité p. 107]
- [139] *CardPeek*. <http://code.google.com/p/cardpeek/>. vu en novembre 2011. [cité p. 107]
- [140] *SmartCard ToolSetPro V3.4 by ScardSoft*. vu en novembre 2011. [cité p. 108]
- [141] *SmartCard framework API by SpringCard*. <http://www.springcard.com/download/-sdks.html>. vu en novembre 2011. [cité p. 108]
- [142] Olivier Rouit. *SmartCard framework API*. vu en novembre 2011. [cité p. 108]
- [143] Licence Code Project Open Licence. <http://www.codeproject.com/info/cpol10.aspx>. [cité p. 108]
- [144] Daniel Müller. *SmartCard Library*. <http://danm.de/index.php?action=source>. [cité p. 108]

- [145] Eduard De Jong, Pieter Hartel, Patrice Peyret, and Peter Cattaneo. Java card : An analysis of the most successful smart card operating system, 2005. [cité p. 109]
- [146] Denis Sabatier and Pierre Lartigue. The Use of the B Formal Method for the Design and the Validation of the Transaction Mechanism for Smart Card Applications. In Jeannette M. Wing, Jim Woodcock, and Jim Davies, editors, *World Congress on Formal Methods*, volume 1708 of *Lecture Notes in Computer Science*, pages 348–368. Springer, 1999. [cité p. 109]
- [147] PC/SC Workgroup, <http://www.pcscworkgroup.com/>. *PC/SC Workgroup Specifications 2*, 2005. [cité p. 109]
- [148] PC/SC Workgroup. <http://www.pcscworkgroup.com/>. [cité p. 110]
- [149] Winscard Module. <http://msdn.microsoft.com/en-us/library/ms924513.aspx>. [cité p. 110]
- [150] MUSCLE - Cross-Platform Smart Card Development. <http://www.musclecard.com/>. [cité p. 110]
- [151] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns*. Addison-Wesley, Boston, MA, January 1995. [cité p. 112]
- [152] ISO/IEC, <http://www.iso.org>. *24727 : Identification cards - Integrated circuit card programming interfaces*. [cité p. 112]
- [153] ISO/IEC, <http://www.iso.org>. *ISO/IEC 7816-1 to 15 : Identification cards - Integrated circuit(s) cards with contacts(Parts 1 to 15)*. [cité p. 112, 116, 127]
- [154] Richard Alleyne. Chip and pin card readers fundamentally flawed. <http://www.telegraph.co.uk/science/science-news/7215920/Chip-and-pin-card-readers-fundamentally-flawed.html>, 2010. [cité p. 120]
- [155] Susan Watts. New flaws in chip and pin system revealed. http://www.bbc.co.uk/blogs/newsnight/susanwatts/2010/02/new_flaws_in_chip_and_pin_syst.html, 2010. [cité p. 120]
- [156] YouTube. Uk chip and pin credit / debit cards are insecure. <http://www.youtube.com/watch?v=JPAX32lgkrw>, 2010. [cité p. 120]
- [157] Julien Lancia. Un framework de fuzzing pour cartes puce : application aux protocoles emv. In *Symposium sur la securit des technologies de l'information et des communications*, 2011. [cité p. 123, 126, 127, 138, 146, 154, 194]
- [158] Mark Dowd, John McDonald, and Justin Schuh. *The Art of Software Security Assessment : Identifying and Preventing Software Vulnerabilities*. Addison-Wesley Professional, 2006. [cité p. 123]

- [159] Gerhard Koning Gans, Jaap-Henk Hoepman, and Flavio D. Garcia. A practical attack on the mifare classic. In *Proceedings of the 8th IFIP WG 8.8/11.2 international conference on Smart Card Research and Advanced Applications*, CARDIS '08, pages 267–282, Berlin, Heidelberg, 2008. Springer-Verlag. [cité p. 124]
- [160] Barton Miller. Cs 736, fall 1988, project list. <http://pages.cs.wisc.edu/~bart/fuzz/CS736-Projects-f1988.pdf>, 1988. [cité p. 124]
- [161] Barton P. Miller, Louis Fredriksen, and Bryan So. An empirical study of the reliability of unix utilities. *Commun. ACM*, 33(12) :32–44, December 1990. [cité p. 124]
- [162] Toby Clarke. Fuzzing for software vulnerability discovery. Technical report, Royal Holloway University of London, 2009. [cité p. 124]
- [163] Jason Crampton Toby Clarke. Fuzzing – or how to help computers cope with the unexpected. Technical report, Royal Holloway University of London. [cité p. 125]
- [164] Vincent Guyot. Smart card, the invisible wallet. In *Proceedings of the 9th European Conference on Information Warfare and Security*, 2010. [cité p. 125]
- [165] Nassima Kamel Matthieu Barraud, Guillaume Bouffard and Jean-Louis Lanet. Fuzzing on the http protocol implementation in mobile embedded web server. In *Proceeding of C&ESAR 2011*, 2011. [cité p. 125]
- [166] Pedram Amini. Sulley fuzzing framework. <http://code.google.com/p/sulley/>. [cité p. 126]
- [167] John H. Holland. *Adaptation in natural and artificial systems*. MIT Press, Cambridge, MA, USA, 1992. [cité p. 130]
- [168] I. Rechenberg. *Evolutionsstrategie*. Frommann-Holzboog-Verlag, 1971. [cité p. 130]
- [169] Matthew Bartschi Wall. *A Genetic Algorithm for Resource-Constrained Scheduling*. PhD thesis, Department of Mechanical Engineering – Massachusetts Institute of Technology, 1996. [cité p. 131]
- [170] Christophe Rosenberger. *Mise en oeuvre d'un système adaptatif de segmentation d'images*. PhD thesis, Université de Rennes 1, 1999. [cité p. 131]
- [171] Aforge.net : : Computer vision, artificial intelligence, robotics. <http://www.aforgenet.com/>. [cité p. 132]
- [172] MasterCard. *M/Chip 4 Version 1.1 - Card Application Specifications for Debit and Credit*. MasterCard, October 2006. [cité p. 132, 141]
- [173] MasterCard. *PayPass - Mag Stripe*. MasterCard, December 2007. [cité p. 139, 140, 195]
- [174] Erik Poll. Openemv. <http://sourceforge.net/projects/openemv/>. [cité p. 151]

- [175] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8(2) :244–263, April 1986. [cité p. 153]
- [176] Irem Aktug and Katsiaryna Naliuka. Conspec a formal language for policy specification. *Electronic Notes in Theoretical Computer Science*, 197(1) :45 – 58, 2008. [cité p. 153]
- [177] Sylvain Vernois Vincent Alimi, Christophe Rosenberger. A mobile contactless point of sale enhanced by the nfc and biometric technologies. *INDERSCIENCE International Journal of Internet Technology and Secured Transactions*, 2012. [cité p. 157]

Annexe

Annexe A

Scripts de déploiement d'un service mobile sans contact

A.1 Récupération des données de configuration du SE – Données vierges

```
INFO 2012-04-05 09:31:24,968 2875ms notifyTransmit - >> 00 A4 04 00 07 A0 00 00
00 AC 10 10
INFO 2012-04-05 09:31:25,046 2953ms notifyTransmit - << 6F 09 84 07 A0 00 00 00
AC 10 10 90-00
INFO 2012-04-05 09:31:25,062 2968ms notifyTransmit - >> 00 A4 00 00 01 01
INFO 2012-04-05 09:31:25,062 2968ms notifyTransmit - << 90-00
INFO 2012-04-05 09:31:25,062 2968ms notifyTransmit - >> 00 B0 00 00 01 00 02
INFO 2012-04-05 09:31:25,062 2968ms notifyTransmit - << 0A AE 90-00
INFO 2012-04-05 09:31:25,109 3015ms notifyTransmit - >> 00 B0 00 02 01 00 FA
INFO 2012-04-05 09:31:25,109 3015ms notifyTransmit - << 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 90-00
```

Listing A.1: Récupération des données de configuration du SE – Données vierges

A.2 Chargement de l'ontologie génération automatique et envoi du script de déploiement

```

Chargement de l'ontologie
-----
INFO 2012-04-05 09:31:35,093 13000ms - Get Individual GlobalPlatformSecureElement :
      gp#gpGlobalPlatformSecureElement
INFO 2012-04-05 09:31:35,093 13000ms - Get Individual GlobalPlatform Registry :
      gp#gpGlobalPlatformRegistry
INFO 2012-04-05 09:31:35,093 13000ms - Get Individual ISD :
      gp#gpIssuerSecurityDomain
INFO 2012-04-05 09:31:35,359 13265ms SetIssuerSecurityDomainParameters -
      Individual AID :
      gp#AID-ISD
INFO 2012-04-05 09:31:35,359 13265ms SetIssuerSecurityDomainParameters -
      DataProperty hasAIDValue :
      gp#hasAIDValue
INFO 2012-04-05 09:31:35,359 13265ms SetIssuerSecurityDomainParameters -
      Statement hasAID :
      [gp#gpIssuerSecurityDomain , gp#hasAID , gp#AID-ISD]
INFO 2012-04-05 09:31:35,359 13265ms SetIssuerSecurityDomainParameters -
      Statement hasApplicationInstanceLifeCycle :
      [gp#gpIssuerSecurityDomain , gp#hasApplicationInstanceLifeCycle ,
      gp#gpLifeCycle_Application\_SD\_PERSONALIZED]
INFO 2012-04-05 09:31:35,375 13281ms SetIssuerSecurityDomainParameters -
      DataProperty hasCardImageNumber :
      gp#hasCardImageNumber
INFO 2012-04-05 09:31:35,375 13281ms SetIssuerSecurityDomainParameters -
      DataProperty hasCarRecognitionData :
      gp#hasCarRecognitionData
INFO 2012-04-05 09:31:35,375 13281ms SetIssuerSecurityDomainParameters -
      DataProperty hasIssuerIdentificationNumber :
      gp#hasIssuerIdentificationNumber

INFO 2012-04-05 09:31:35,417 13000ms LaunchReasoner - Launching reasoner
INFO 2012-04-05 09:31:35,703 13250ms - Initialization time: 843ms

-----
Prise de décision par le framework et génération du script de déploiement
-----
INFO 2012-04-05 09:31:37,484 15390ms GenerateDeploymentScenario - An Executable
      Load File containing the Executable Module A0000000041010 must be loaded onto
      the SE
INFO 2012-04-05 09:31:37,484 15390ms GenerateDeploymentScenario - The instance
      A0000000041010 of the Executable Module A0000000041010 will be installed from
      the elf

-----
Envoi du script de déploiement au SE
-----
INFO 2012-04-05 09:31:38,234 16140ms notifyTransmit - >> 00 A4 04 00 08 A0 00 00
      00 03 00 00 00 00

```

INFO 2012-04-05 09:31:38,234 16140ms notifyTransmit - << 6F 10 84 08 A0 00 00 00 03 00 00 00 A5 04 9F 65 01 FF 90-00

[Initialisation du canal sécurisé]

INFO 2012-04-05 09:31:38,296 16203ms notifyTransmit - >> 80 50 00 00 08 11 22 33 44 55 66 77 88 00

INFO 2012-04-05 09:31:38,296 16203ms notifyTransmit - << 00 00 10 23 8F 77 1D 80 21 17 04 02 00 A3 8E 23 8E 45 F7 D1 27 C0 2D 8A E7 21 56 B7 90-00

INFO 2012-04-05 09:31:38,468 16375ms notifyTransmit - >> 84 82 01 00 10 29 6F 3C AD 86 04 7A 7D 19 77 F6 F8 72 42 CC 4E 00

INFO 2012-04-05 09:31:38,484 16390ms notifyTransmit - << 90-00

[INSTALL for load]

INFO 2012-04-05 09:31:39,437 17343ms notifyTransmit - >> 84 E6 02 00 1A 05 A0 00 00 00 04 08 A0 00 00 00 03 00 00 00 00 00 00 00 6D 8C FF 49 24 6C F6 1A 00

INFO 2012-04-05 09:31:39,437 17343ms notifyTransmit - << 00 90-00

[LOAD #1]

INFO 2012-04-05 09:31:39,796 17703ms notifyTransmit - >> 84 E8 00 00 F7 C4 82 14 55 01 00 38 DE CA FF ED 02 02 04 00 01 05 A0 00 00 00 04 28 63 6F 6D 2F 69 6E 73 69 64 65 63 6F 6E 74 61 63 74 6C 65 73 73 2F 6D 61 73 74 65 72 63 61 72 64 2F 70 61 79 70 61 73 73 02 00 21 00 38 00 21 00 0B 00 32 01 A6 00 38 0B B9 00 48 01 97 00 00 04 2B 2D D6 00 14 00 04 00 30 05 01 00 04 00 32 05 00 01 07 A0 00 00 00 62 00 01 00 01 07 A0 00 00 00 62 01 01 00 01 06 A0 00 00 01 51 00 01 01 07 A0 00 00 00 62 01 02 01 01 07 A0 00 00 00 62 02 01 03 00 0B 01 07 A0 00 00 00 04 10 10 00 58 06 00 38 00 00 00 80 00 00 FF 00 01 00 00 00 00 81 03 0C 00 0A 04 05 00 00 00 76 FF FF 00 6C 07 B1 04 47 00 80 00 03 00 02 01 00 00 07 0A 28 0A 66 0A 87 0A A1 0A B7 0A CF 0A FF 07 0B B9 0A 02 79 80 0E 02 89 00 17 03 10 22 4F 33 20 FC E7 01 81 00

INFO 2012-04-05 09:31:39,796 17703ms notifyTransmit - << 00 90-00

[LOAD #2]

INFO 2012-04-05 09:31:39,890 17796ms notifyTransmit - >> 84 E8 00 01 F7 00 1C 03 2E 00 24 03 10 00 1C 03 31 00 25 03 10 80 1C 03 34 00 26 03 C8 80 12 03 DC 00 2B 04 24 80 12 04 38 00 2B 04 E2 80 2B 05 0F 00 2F 07 10 80 12 07 22 00 00 09 6B 80 12 09 83 00 26 0B 2D 80 2F 0B 5C 00 00 01 10 18 8C 00 1F 7A 05 30 8F 00 39 3D 8C 00 58 18 1D 04 41 18 1D 25 8B 00 15 7A 02 10 18 8D 00 18 87 00 04 78 01 10 AD 00 8E 01 00 22 05 7A 05 22 19 8B 00 27 2D 1A 03 25 10 80 6A 08 11 68 00 8D 00 32 AD 01 05 25 04 6A 08 11 69 85 8D 00 32 1A 05 8D 00 3E 32 1F 73 00 21 00 CF 00 CF 00 09 1A 03 10 CF 38 1A 04 10 0B 38 7B 00 40 03 1A 05 10 0B 8D 00 45 3B 70 08 11 6A 88 8D 00 32 19 03 10 0D 8B 00 57 7A 05 21 19 8B 00 27 2D 1A 03 25 10 80 6A 08 11 6E 00 8D 00 32 1A 05 25 61 07 1A 06 25 60 08 11 6A 86 8D 00 99 68 8D 1B 45 5B 23 A5 00

INFO 2012-04-05 09:31:39,906 17812ms notifyTransmit - << 00 90-00

.....
.....
.....
.....
.....

[LOAD #21]

INFO 2012-04-05 09:31:43,390 21296ms notifyTransmit - >> 84 E8 80 15 C6 01 54 00 E4 00 D8 01 58 01 2E 01 5E 00 EF 00 EF 00 EF 00 EF 00 D8 01 63 01 67 01 1B 01 6B 05 68 20 20 01 B0 05 60 02 00 05 E0 02 00 01 40 05 68 30 A0 01 30 01 C0 04 B4 34 01 10 03 33 40 02 44 03 BB 20 04 B4 31 0B 68 10 A6 00 20 10 05 68 10 60 03 B4 30 06 B4 B4 43 01 20 06 B4 44 34 04 4B 41 06 B4 B4 44 04 B4 41 02 33 04 B4 44 05 68 40 10 02 41 07 68 30 03 10 06 B4 4B 44 05 B4 43 40 02 31 03 B4 40


```

03 B4 10 03 43 B0 07 32 68 40 10 08 34 26 83 00 06 68 10 A1 02 32 03 44 10 05
68 30 F0 0A 68 30 03 B4 41 07 B4 4B 44 20 06 36 83 0E 05 68 30 E0 07 32 68 30
F0 05 B4 44 10 88 36 88 77 51 71 1D C3 00
INFO 2012-04-05 09:31:43,390 21296ms notifyTransmit - << 00 90-00

[INSTALL for install de l'instance A0 00 00 00 04 10 10]
INFO 2012-04-05 09:31:44,062 21968ms notifyTransmit - >> 84 E6 0C 00 24 05 A0 00
00 00 04 07 A0 00 00 00 04 10 10 07 A0 00 00 00 04 10 10 01 00 02 C9 00 00 D4
B1 CF 9D 5C 53 B6 A0 00
INFO 2012-04-05 09:31:44,062 21968ms notifyTransmit - << 00 90-00

-----
Personnalisation de l'instance
-----
INFO 2012-04-05 09:31:44,328 22234ms notifyTransmit - >> 00 A4 04 00 07 A0 00 00
00 04 10 10 00
INFO 2012-04-05 09:31:44,328 22234ms notifyTransmit - << 6F 17 84 07 A0 00 00 00
04 10 10 A5 0C 50 0A 4D 61 73 74 65 72 43 61 72 64 90-00
INFO 2012-04-05 09:31:44,343 22250ms notifyTransmit - >> 80 50 04 00 08 11 22 33
44 55 66 77 88 00
INFO 2012-04-05 09:31:44,343 22250ms notifyTransmit - << 6A-81
INFO 2012-04-05 09:31:44,593 22500ms notifyTransmit - >> 84 E2 A0 00 B3 01 01 7F
9F 6C 02 00 01 56 3E 42 35 34 31 33 31 32 33 34 35 36 37 38 34 38 30 30 5E 53
55 50 50 4C 49 45 44 2F 4E 4F 54 5E 30 39 30 36 31 30 31 33 33 30 30 30 33 33
33 30 30 30 32 32 32 32 32 30 30 30 31 31 31 31 30 9F 64 01 03 9F 62 06 00 00
00 38 00 00 9F 63 06 00 00 00 00 E0 E0 9F 65 02 00 0E 9F 66 02 0E 70 9F 6B 13
54 13 12 34 56 78 48 00 D0 90 61 01 90 00 99 00 00 00 0F 9F 67 01 03 A0 01 0B
00 00 40 00 00 00 00 77 80 99 D3 A0 02 10 52 29 A2 B1 82 0F 32 13 CA F2 24 3C
B1 9C 5D F7 DE 65 E2 9F 48 C7 F2 12 53 61 65 BE EF EE 57 06
INFO 2012-04-05 09:31:44,593 22500ms notifyTransmit - << 90-00

```

Listing A.2: Chargement de l'ontologie génération automatique et envoi du script de déploiement

A.3 Mise à jour des données de configuration dans l'applet

```

-----
Sélection de l'applet de configuration
-----
INFO 2012-04-05 09:31:46,421 24328ms notifyTransmit - >> 00 A4 04 00 07 A0 00 00
00 AC 10 10 00
INFO 2012-04-05 09:31:46,421 24328ms notifyTransmit - << 90-00

-----
Sélection du fichier
-----
INFO 2012-04-05 09:31:46,421 24328ms notifyTransmit - >> 00 A4 00 00 01 01
INFO 2012-04-05 09:31:46,421 24328ms notifyTransmit - << 90-00

```

Ecriture des données de configuration

```

INFO 2012-04-05 09:31:46,515 24421ms notifyTransmit - >> 00 D0 00 00 FA 0A AE 20
 20 20 20 3C 21 2D 2D 20 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69
 63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F 32
 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 65 6C 66 4D 43 20
 2D 2D 3E 0D 0A 0D 0A 20 20 20 20 3C 6F 77 6C 3A 4E 61 6D 65 64 49 6E 64 69 76
 69 64 75 61 6C 20 72 64 66 3A 61 62 6F 75 74 3D 22 68 74 74 70 3A 2F 2F 77 77
 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67 69
 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E 6F
 77 6C 23 65 6C 66 4D 43 22 3E 0D 0A 20 20 20 20 20 20 20 20 3C 72 64 66 3A 74
 79 70 65 20 72 64 66 3A 72 65 73 6F 75 72 63 65 3D 22 68 74 74 70 3A 2F 2F 77
 77 77 2E 73 65 6D 61 6E 74 69 63 77 65

INFO 2012-04-05 09:31:46,531 24437ms notifyTransmit - << 90-00

INFO 2012-04-05 09:31:46,609 24515ms notifyTransmit - >> 00 D0 00 FA FA 62 2E 6F
 72 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61
 6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 45 78 65 63 75 74 61 62 6C 65 4C 6F
 61 64 46 69 6C 65 22 2F 3E 0D 0A 20 20 20 20 20 20 20 20 3C 68 61 73 41 49 44
 20 72 64 66 3A 72 65 73 6F 75 72 63 65 3D 22 68 74 74 70 3A 2F 2F 77 77 77 2E
 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73
 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C
 23 65 6C 66 4D 43 41 69 64 22 2F 3E 0D 0A 20 20 20 20 20 20 20 20 3C 68 61 73
 45 78 65 63 75 74 61 62 6C 65 4D 6F 64 75 6C 65 20 72 64 66 3A 72 65 73 6F 75
 72 63 65 3D 22 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69 63 77 65
 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67

INFO 2012-04-05 09:31:46,625 24531ms notifyTransmit - << 90-00

INFO 2012-04-05 09:31:46,718 24625ms notifyTransmit - >> 00 D0 01 F4 FA 69 65 73
 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C
 23 65 6D 4D 43 22 2F 3E 0D 0A 20 20 20 20 3C 2F 6F 77 6C 3A 4E 61 6D 65 64 49
 6E 64 69 76 69 64 75 61 6C 3E 0D 0A 20 20 20 20 0D 0A 20 20 20 20 3C 21 2D 2D
 20 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72
 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C
 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 65 6C 66 4D 43 41 69 64 20 2D 2D 3E 0D
 0A 0D 0A 20 20 20 20 3C 6F 77 6C 3A 4E 61 6D 65 64 49 6E 64 69 76 69 64 75 61
 6C 20 72 64 66 3A 61 62 6F 75 74 3D 22 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65
 6D 61 6E 74 69 63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32
 30 31 31 2F 32 2F 47 6C 6F 62 61 6C 50

INFO 2012-04-05 09:31:46,718 24625ms notifyTransmit - << 90-00

INFO 2012-04-05 09:31:46,812 24718ms notifyTransmit - >> 00 D0 02 EE FA 6C 61 74
 66 6F 72 6D 2E 6F 77 6C 23 65 6C 66 4D 43 41 69 64 22 3E 0D 0A 20 20 20 20 20
 20 20 20 3C 72 64 66 3A 74 79 70 65 20 72 64 66 3A 72 65 73 6F 75 72 63 65 3D
 22 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72
 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C
 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 41 49 44 22 2F 3E 0D 0A 20 20 20 20 20
 20 20 20 3C 68 61 73 41 49 44 56 61 6C 75 65 3E 41 30 30 30 30 30 30 30 30 34
 3C 2F 68 61 73 41 49 44 56 61 6C 75 65 3E 0D 0A 20 20 20 20 3C 2F 6F 77 6C 3A
 4E 61 6D 65 64 49 6E 64 69 76 69 64 75 61 6C 3E 0D 0A 20 20 20 20 0D 0A 20 20
 20 20 3C 21 2D 2D 20 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69 63
 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C

INFO 2012-04-05 09:31:46,828 24734ms notifyTransmit - << 90-00

INFO 2012-04-05 09:31:46,906 24812ms notifyTransmit - >> 00 D0 03 E8 FA 6F 67 69
 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E 6F
 77 6C 23 65 6D 4D 43 20 2D 2D 3E 0D 0A 0D 0A 20 20 20 20 3C 6F 77 6C 3A 4E 61
 6D 65 64 49 6E 64 69 76 69 64 75 61 6C 20 72 64 66 3A 61 62 6F 75 74 3D 22 68
 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72 67 2F
 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C 50 6C
 61 74 66 6F 72 6D 2E 6F 77 6C 23 65 6D 4D 43 22 3E 0D 0A 20 20 20 20 20 20

```

ANNEXE A. SCRIPTS DE DÉPLOIEMENT D'UN SERVICE MOBILE SANS CONTACT

180

```

20 3C 72 64 66 3A 74 79 70 65 20 72 64 66 3A 72 65 73 6F 75 72 63 65 3D 22 68
74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72 67 2F
6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C 50 6C
61 74 66 6F 72 6D 2E 6F 77 6C 23 45 78
INFO 2012-04-05 09:31:46,921 24828ms notifyTransmit - << 90-00
INFO 2012-04-05 09:31:47,015 24921ms notifyTransmit - >> 00 D0 04 E2 FA 65 63 75
74 61 62 6C 65 4D 6F 64 75 6C 65 22 2F 3E 0D 0A 20 20 20 20 20 20 20 20 3C 68
61 73 41 49 44 20 72 64 66 3A 72 65 73 6F 75 72 63 65 3D 22 68 74 74 70 3A 2F
2F 77 77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C
6F 67 69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72
6D 2E 6F 77 6C 23 65 6D 4D 43 41 69 64 22 2F 3E 0D 0A 20 20 20 20 3C 2F 6F 77
6C 3A 4E 61 6D 65 64 49 6E 64 69 76 69 64 75 61 6C 3E 0D 0A 20 20 20 20 0D 0A
20 20 20 20 3C 21 2D 2D 20 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74
69 63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F
32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 65 6D 4D 43 41
69 64 20 2D 2D 3E 0D 0A 0D 0A 20 20 20
INFO 2012-04-05 09:31:47,015 24921ms notifyTransmit - << 90-00
INFO 2012-04-05 09:31:47,109 25015ms notifyTransmit - >> 00 D0 05 DC FA 20 3C 6F
77 6C 3A 4E 61 6D 65 64 49 6E 64 69 76 69 64 75 61 6C 20 72 64 66 3A 61 62 6F
75 74 3D 22 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62
2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F
6A 61 6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 65 6D 4D 43 41 69 64 22 3E 0D
0A 20 20 20 20 20 20 20 20 3C 72 64 66 3A 74 79 70 65 20 72 64 66 3A 72 65 73
6F 75 72 63 65 3D 22 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69 63
77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F 32 2F
47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 41 49 44 22 2F 3E 0D
0A 20 20 20 20 20 20 20 20 3C 68 61 73 41 49 44 56 61 6C 75 65 3E 41 30 30 30
30 30 30 30 30 34 31 30 31 30 3C 2F 68
INFO 2012-04-05 09:31:47,109 25015ms notifyTransmit - << 90-00
INFO 2012-04-05 09:31:47,203 25109ms notifyTransmit - >> 00 D0 06 D6 FA 61 73 41
49 44 56 61 6C 75 65 3E 0D 0A 20 20 20 20 3C 2F 6F 77 6C 3A 4E 61 6D 65 64 49
6E 64 69 76 69 64 75 61 6C 3E 0D 0A 0D 0A 20 20 20 20 3C 21 2D 2D 20 68 74 74
70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72 67 2F 6F 6E
74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C 50 6C 61 74
66 6F 72 6D 2E 6F 77 6C 23 61 70 70 4D 43 41 69 64 20 2D 2D 3E 0D 0A 0D 0A 20
20 20 20 3C 6F 77 6C 3A 4E 61 6D 65 64 49 6E 64 69 76 69 64 75 61 6C 20 72 64
66 3A 61 62 6F 75 74 3D 22 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74
69 63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F
32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 61 70 70 4D 43
41 69 64 22 3E 0D 0A 09 3C 72 64 66 3A
INFO 2012-04-05 09:31:47,203 25109ms notifyTransmit - << 90-00
INFO 2012-04-05 09:31:47,296 25203ms notifyTransmit - >> 00 D0 07 D0 FA 74 79 70
65 20 72 64 66 3A 72 65 73 6F 75 72 63 65 3D 22 68 74 74 70 3A 2F 2F 77 77 77
2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67 69 65
73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E 6F 77
6C 23 41 49 44 22 2F 3E 0D 0A 09 3C 68 61 73 41 49 44 56 61 6C 75 65 3E 41 30
30 30 30 30 30 30 30 34 31 30 31 30 3C 2F 68 61 73 41 49 44 56 61 6C 75 65 3E
0D 0A 20 20 20 20 3C 2F 6F 77 6C 3A 4E 61 6D 65 64 49 6E 64 69 76 69 64 75 61
6C 3E 0D 0A 0D 0A 20 20 20 20 3C 21 2D 2D 20 68 74 74 70 3A 2F 2F 77 77 77 2E
73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73
2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C
23 6D 63 41 70 70 20 2D 2D 3E 0D 0A 0D
INFO 2012-04-05 09:31:47,296 25203ms notifyTransmit - << 90-00
INFO 2012-04-05 09:31:47,390 25296ms notifyTransmit - >> 00 D0 08 CA FA 0A 20 20
20 20 3C 6F 77 6C 3A 4E 61 6D 65 64 49 6E 64 69 76 69 64 75 61 6C 20 72 64 66
3A 61 62 6F 75 74 3D 22 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69
63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F 32

```

```

2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 6D 63 41 70 70 22
3E 0D 0A 09 3C 72 64 66 3A 74 79 70 65 20 72 64 66 3A 72 65 73 6F 75 72 63 65
3D 22 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F
72 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61
6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 41 70 70 6C 69 63 61 74 69 6F 6E 49
6E 73 74 61 6E 63 65 22 2F 3E 0D 0A 09 3C 68 61 73 41 49 44 20 72 64 66 3A 72
65 73 6F 75 72 63 65 3D 22 68 74 74 70
INFO 2012-04-05 09:31:47,406 25312ms notifyTransmit - << 90-00
INFO 2012-04-05 09:31:47,484 25390ms notifyTransmit - >> 00 D0 09 C4 EC 3A 2F 2F
77 77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F
67 69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D
2E 6F 77 6C 23 61 70 70 4D 43 41 69 64 22 2F 3E 0D 0A 09 3C 68 61 73 41 73 73
6F 63 69 61 74 65 64 53 65 63 75 72 69 74 79 44 6F 6D 61 69 6E 20 72 64 66 3A
72 65 73 6F 75 72 63 65 3D 22 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E
74 69 63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31
2F 32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 67 70 49 73
73 75 65 72 53 65 63 75 72 69 74 79 44 6F 6D 61 69 6E 22 2F 3E 0D 0A 20 20 20
20 3C 2F 6F 77 6C 3A 4E 61 6D 65 64 49 6E 64 69 76 69 64 75 61 6C 3E 0D 0A
INFO 2012-04-05 09:31:47,484 25390ms notifyTransmit - << 90-00

```

Listing A.3: Mise à jour des données de configuration dans l'applet

A.4 Récupération des données de configuration depuis l'applet chargement dans le framework et requête identique de déploiement

```

-----
Sélection de l'applet de configuration
-----
INFO 2012-04-05 09:31:50,390 28296ms notifyTransmit - >> 00 A4 04 00 07 A0 00 00
00 AC 10 10 00
INFO 2012-04-05 09:31:50,390 28296ms notifyTransmit - << 90-00
-----

Sélection du fichier
-----
INFO 2012-04-05 09:31:50,390 28296ms notifyTransmit - >> 00 A4 00 00 01 01
INFO 2012-04-05 09:31:50,406 28312ms notifyTransmit - << 90-00
-----

Ecriture des données de configuration
-----
INFO 2012-04-05 09:31:50,406 28312ms notifyTransmit - >> 00 B0 00 00 01 00 02
INFO 2012-04-05 09:31:50,406 28312ms notifyTransmit - << 0A AE 90-00
INFO 2012-04-05 09:31:50,453 28359ms notifyTransmit - >> 00 B0 00 02 01 00 FA
INFO 2012-04-05 09:31:50,453 28359ms notifyTransmit - << 20 20 20 20 3C 21 2D 2D
20 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72
67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C
50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 65 6C 66 4D 43 20 2D 2D 3E 0D 0A 0D 0A
20 20 20 20 3C 6F 77 6C 3A 4E 61 6D 65 64 49 6E 64 69 76 69 64 75 61 6C 20 72

```

```

64 66 3A 61 62 6F 75 74 3D 22 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E
74 69 63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31
2F 32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 65 6C 66 4D
43 22 3E 0D 0A 20 20 20 20 20 20 20 20 20 3C 72 64 66 3A 74 79 70 65 20 72 64 66
3A 72 65 73 6F 75 72 63 65 3D 22 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61
6E 74 69 63 77 65 62 2E 90-00
INFO 2012-04-05 09:31:50,500 28406ms notifyTransmit - >> 00 B0 00 FC 01 00 FA
INFO 2012-04-05 09:31:50,500 28406ms notifyTransmit - << 6F 72 67 2F 6F 6E 74 6F
6C 6F 67 69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F
72 6D 2E 6F 77 6C 23 45 78 65 63 75 74 61 62 6C 65 4C 6F 61 64 46 69 6C 65 22
2F 3E 0D 0A 20 20 20 20 20 20 20 20 20 3C 68 61 73 41 49 44 20 72 64 66 3A 72 65
73 6F 75 72 63 65 3D 22 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69
63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F 32
2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 65 6C 66 4D 43 41
69 64 22 2F 3E 0D 0A 20 20 20 20 20 20 20 20 20 3C 68 61 73 45 78 65 63 75 74 61
62 6C 65 4D 6F 64 75 6C 65 20 72 64 66 3A 72 65 73 6F 75 72 63 65 3D 22 68 74
74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72 67 2F 6F
6E 74 6F 6C 6F 67 69 65 90-00
INFO 2012-04-05 09:31:50,546 28453ms notifyTransmit - >> 00 B0 01 F6 01 00 FA
INFO 2012-04-05 09:31:50,546 28453ms notifyTransmit - << 73 2F 32 30 31 31 2F 32
2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 65 6D 4D 43 22 2F
3E 0D 0A 20 20 20 20 20 20 3C 2F 6F 77 6C 3A 4E 61 6D 65 64 49 6E 64 69 76 69 64 75
61 6C 3E 0D 0A 20 20 20 20 0D 0A 20 20 20 3C 21 2D 2D 20 68 74 74 70 3A 2F
2F 77 77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C
6F 67 69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72
6D 2E 6F 77 6C 23 65 6C 66 4D 43 41 69 64 20 2D 2D 3E 0D 0A 0D 0A 20 20 20 20
3C 6F 77 6C 3A 4E 61 6D 65 64 49 6E 64 69 76 69 64 75 61 6C 20 72 64 66 3A 61
62 6F 75 74 3D 22 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69 63 77
65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F 32 2F 47
6C 6F 62 61 6C 50 6C 61 90-00
INFO 2012-04-05 09:31:50,609 28515ms notifyTransmit - >> 00 B0 02 F0 01 00 FA
INFO 2012-04-05 09:31:50,609 28515ms notifyTransmit - << 74 66 6F 72 6D 2E 6F 77
6C 23 65 6C 66 4D 43 41 69 64 22 3E 0D 0A 20 20 20 20 20 20 20 20 20 3C 72 64 66
3A 74 79 70 65 20 72 64 66 3A 72 65 73 6F 75 72 63 65 3D 22 68 74 74 70 3A 2F
2F 77 77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C
6F 67 69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72
6D 2E 6F 77 6C 23 41 49 44 22 2F 3E 0D 0A 20 20 20 20 20 20 20 20 20 3C 68 61 73
41 49 44 56 61 6C 75 65 3E 41 30 30 30 30 30 30 30 30 34 3C 2F 68 61 73 41 49
44 56 61 6C 75 65 3E 0D 0A 20 20 20 20 3C 2F 6F 77 6C 3A 4E 61 6D 65 64 49 6E
64 69 76 69 64 75 61 6C 3E 0D 0A 20 20 20 20 0D 0A 20 20 20 20 3C 21 2D 2D 20
68 74 74 70 3A 2F 2F 77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72 67
2F 6F 6E 74 6F 6C 6F 67 90-00
INFO 2012-04-05 09:31:50,656 28562ms notifyTransmit - >> 00 B0 03 EA 01 00 FA
INFO 2012-04-05 09:31:50,656 28562ms notifyTransmit - << 69 65 73 2F 32 30 31 31
2F 32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 65 6D 4D 43
20 2D 2D 3E 0D 0A 0D 0A 20 20 20 20 3C 6F 77 6C 3A 4E 61 6D 65 64 49 6E 64 69
76 69 64 75 61 6C 20 72 64 66 3A 61 62 6F 75 74 3D 22 68 74 74 70 3A 2F 2F 77
77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67
69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E
6F 77 6C 23 65 6D 4D 43 22 3E 0D 0A 20 20 20 20 20 20 20 20 20 3C 72 64 66 3A 74
79 70 65 20 72 64 66 3A 72 65 73 6F 75 72 63 65 3D 22 68 74 74 70 3A 2F 2F 77
77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67
69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E
6F 77 6C 23 45 78 65 63 90-00
INFO 2012-04-05 09:31:50,703 28609ms notifyTransmit - >> 00 B0 04 E4 01 00 FA
INFO 2012-04-05 09:31:50,703 28609ms notifyTransmit - << 75 74 61 62 6C 65 4D 6F
64 75 6C 65 22 2F 3E 0D 0A 20 20 20 20 20 20 20 20 20 3C 68 61 73 41 49 44 20 72

```

A.4. RÉCUPÉRATION DES DONNÉES DE CONFIGURATION DEPUIS L'APPLET
 CHARGEMENT DANS LE FRAMEWORK ET REQUÊTE IDENTIQUE DE
 DÉPLOIEMENT

```

64 66 3A 72 65 73 6F 75 72 63 65 3D 22 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65
6D 61 6E 74 69 63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32
30 31 31 2F 32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 65
6D 4D 43 41 69 64 22 2F 3E 0D 0A 20 20 20 20 3C 2F 6F 77 6C 3A 4E 61 6D 65 64
49 6E 64 69 76 69 64 75 61 6C 3E 0D 0A 20 20 20 20 0D 0A 20 20 20 20 3C 21 2D
2D 20 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F
72 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61
6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 65 6D 4D 43 41 69 64 20 2D 2D 3E 0D
0A 0D 0A 20 20 20 20 3C 90-00
INFO 2012-04-05 09:31:50,750 28656ms notifyTransmit - >> 00 B0 05 DE 01 00 FA
INFO 2012-04-05 09:31:50,765 28671ms notifyTransmit - << 6F 77 6C 3A 4E 61 6D 65
64 49 6E 64 69 76 69 64 75 61 6C 20 72 64 66 3A 61 62 6F 75 74 3D 22 68 74 74
70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72 67 2F 6F 6E
74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C 50 6C 61 74
66 6F 72 6D 2E 6F 77 6C 23 65 6D 4D 43 41 69 64 22 3E 0D 0A 20 20 20 20 20 20
20 20 3C 72 64 66 3A 74 79 70 65 20 72 64 66 3A 72 65 73 6F 75 72 63 65 3D 22
68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72 67
2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C 50
6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 41 49 44 22 2F 3E 0D 0A 20 20 20 20 20 20
20 20 3C 68 61 73 41 49 44 56 61 6C 75 65 3E 41 30 30 30 30 30 30 30 30 34 31
30 31 30 3C 2F 68 61 73 90-00
INFO 2012-04-05 09:31:50,812 28718ms notifyTransmit - >> 00 B0 06 D8 01 00 FA
INFO 2012-04-05 09:31:50,812 28718ms notifyTransmit - << 41 49 44 56 61 6C 75 65
3E 0D 0A 20 20 20 20 3C 2F 6F 77 6C 3A 4E 61 6D 65 64 49 6E 64 69 76 69 64 75
61 6C 3E 0D 0A 0D 0A 20 20 20 20 3C 21 2D 2D 20 68 74 74 70 3A 2F 2F 77 77 77
2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67 69 65
73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E 6F 77
6C 23 61 70 70 4D 43 41 69 64 20 2D 2D 3E 0D 0A 0D 0A 20 20 20 20 3C 6F 77 6C
3A 4E 61 6D 65 64 49 6E 64 69 76 69 64 75 61 6C 20 72 64 66 3A 61 62 6F 75 74
3D 22 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F
72 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61
6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 61 70 70 4D 43 41 69 64 22 3E 0D 0A
09 3C 72 64 66 3A 74 79 90-00
INFO 2012-04-05 09:31:50,859 28765ms notifyTransmit - >> 00 B0 07 D2 01 00 FA
INFO 2012-04-05 09:31:50,859 28765ms notifyTransmit - << 70 65 20 72 64 66 3A 72
65 73 6F 75 72 63 65 3D 22 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74
69 63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F
32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 41 49 44 22 2F
3E 0D 0A 09 3C 68 61 73 41 49 44 56 61 6C 75 65 3E 41 30 30 30 30 30 30 30 30
34 31 30 31 30 3C 2F 68 61 73 41 49 44 56 61 6C 75 65 3E 0D 0A 20 20 20 20 3C
2F 6F 77 6C 3A 4E 61 6D 65 64 49 6E 64 69 76 69 64 75 61 6C 3E 0D 0A 0D 0A 20
20 20 20 3C 21 2D 2D 20 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69
63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F 32
2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 6D 63 41 70 70 20
2D 2D 3E 0D 0A 0D 0A 20 90-00
INFO 2012-04-05 09:31:50,906 28812ms notifyTransmit - >> 00 B0 08 CC 01 00 FA
INFO 2012-04-05 09:31:50,906 28812ms notifyTransmit - << 20 20 20 3C 6F 77 6C 3A
4E 61 6D 65 64 49 6E 64 69 76 69 64 75 61 6C 20 72 64 66 3A 61 62 6F 75 74 3D
22 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72
67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C
50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 6D 63 41 70 70 22 3E 0D 0A 09 3C 72 64
66 3A 74 79 70 65 20 72 64 66 3A 72 65 73 6F 75 72 63 65 3D 22 68 74 74 70 3A
2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F
6C 6F 67 69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F
72 6D 2E 6F 77 6C 23 41 70 70 6C 69 63 61 74 69 6F 6E 49 6E 73 74 61 6E 63 65
22 2F 3E 0D 0A 09 3C 68 61 73 41 49 44 20 72 64 66 3A 72 65 73 6F 75 72 63 65
3D 22 68 74 74 70 3A 2F 90-00

```

```
INFO 2012-04-05 09:31:50,953 28859ms notifyTransmit - >> 00 B0 09 C6 01 00 E8
INFO 2012-04-05 09:31:50,953 28859ms notifyTransmit - << 2F 77 77 77 2E 73 65 6D
61 6E 74 69 63 77 65 62 2E 6F 72 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30
31 31 2F 32 2F 47 6C 6F 62 61 6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 61 70
70 4D 43 41 69 64 22 2F 3E 0D 0A 09 3C 68 61 73 41 73 73 6F 63 69 61 74 65 64
53 65 63 75 72 69 74 79 44 6F 6D 61 69 6E 20 72 64 66 3A 72 65 73 6F 75 72 63
65 3D 22 68 74 74 70 3A 2F 2F 77 77 77 2E 73 65 6D 61 6E 74 69 63 77 65 62 2E
6F 72 67 2F 6F 6E 74 6F 6C 6F 67 69 65 73 2F 32 30 31 31 2F 32 2F 47 6C 6F 62
61 6C 50 6C 61 74 66 6F 72 6D 2E 6F 77 6C 23 67 70 49 73 73 75 65 72 53 65 63
75 72 69 74 79 44 6F 6D 61 69 6E 22 2F 3E 0D 0A 20 20 20 20 3C 2F 6F 77 6C 3A
4E 61 6D 65 64 49 6E 64 69 76 69 64 75 61 6C 3E 90-00
```

Affichage des données de configuration

```
<!-- gp#elfMC -->
<owl:NamedIndividual rdf:about="gp#elfMC">
  <rdf:type rdf:resource="gp#ExecutableLoadFile"/>
  <hasAID rdf:resource="gp#elfMCAid"/>
  <hasExecutableModule rdf:resource="gp#emMC"/>
</owl:NamedIndividual>

<!-- gp#elfMCAid -->
<owl:NamedIndividual rdf:about="gp#elfMCAid">
  <rdf:type rdf:resource="gp#AID"/>
  <hasAIDValue>A000000004</hasAIDValue>
</owl:NamedIndividual>

<!-- gp#emMC -->
<owl:NamedIndividual rdf:about="gp#emMC">
  <rdf:type rdf:resource="gp#ExecutableModule"/>
  <hasAID rdf:resource="gp#emMCAid"/>
</owl:NamedIndividual>

<!-- gp#emMCAid -->
<owl:NamedIndividual rdf:about="gp#emMCAid">
  <rdf:type rdf:resource="gp#AID"/>
  <hasAIDValue>A0000000041010</hasAIDValue>
</owl:NamedIndividual>

<!-- gp#appMCAid -->
<owl:NamedIndividual rdf:about="gp#appMCAid">
  <rdf:type rdf:resource="gp#AID"/>
  <hasAIDValue>A0000000041010</hasAIDValue>
</owl:NamedIndividual>

<!-- gp#mcApp -->
<owl:NamedIndividual rdf:about="gp#mcApp">
  <rdf:type rdf:resource="gp#ApplicationInstance"/>
  <hasAID rdf:resource="gp#appMCAid"/>
  <hasAssociatedSecurityDomain
    rdf:resource="gp#gpIssuerSecurityDomain"/>
</owl:NamedIndividual>
```

Chargement de l'ontologie enrichie

A.4. RÉCUPÉRATION DES DONNÉES DE CONFIGURATION DEPUIS L'APPLET
CHARGEMENT DANS LE FRAMEWORK ET REQUÊTE IDENTIQUE DE
DÉPLOIEMENT

185

```
INFO 2012-04-05 09:31:52,453 30359ms - Get Individual GlobalPlatformSecureElement :
gp#gpGlobalPlatformSecureElement
INFO 2012-04-05 09:31:52,453 30359ms - Get Individual GlobalPlatform Registry :
gp#gpGlobalPlatformRegistry
INFO 2012-04-05 09:31:52,453 30359ms - Get Individual ISD :
gp#gpIssuerSecurityDomain
INFO 2012-04-05 09:31:52,625 30531ms SetIssuerSecurityDomainParameters -
Individual AID :
gp#AID-ISD
INFO 2012-04-05 09:31:52,625 30531ms SetIssuerSecurityDomainParameters -
DataProperty hasAIDValue :
gp#hasAIDValue
INFO 2012-04-05 09:31:52,625 30531ms SetIssuerSecurityDomainParameters -
Statement hasAID :
[gp#gpIssuerSecurityDomain , gp#hasAID , gp#AID-ISD]
INFO 2012-04-05 09:31:52,625 30531ms SetIssuerSecurityDomainParameters -
Statement hasApplicationInstanceLifeCycle :
[gp#gpIssuerSecurityDomain , gp#hasApplicationInstanceLifeCycle ,
gp#gpLifeCycle\_Application\_SD\_PERSONALIZED]
INFO 2012-04-05 09:31:52,640 30546ms SetIssuerSecurityDomainParameters -
DataProperty hasCardImageNumber :
gp#hasCardImageNumber
INFO 2012-04-05 09:31:52,640 30546ms SetIssuerSecurityDomainParameters -
DataProperty hasCarRecognitionData :
gp#hasCarRecognitionData
INFO 2012-04-05 09:31:52,640 30546ms SetIssuerSecurityDomainParameters -
DataProperty hasIssuerIdentificationNumber :
gp#hasIssuerIdentificationNumber

INFO 2012-04-05 09:31:52,768 30375ms LaunchReasoner - Launching reasoner
INFO 2012-04-05 09:31:52,925 30531ms - Initialization time: 734ms
```

Prise de décision par le framework et génération du script de déploiement

```
INFO 2012-04-05 09:31:53,765 31671ms GenerateDeploymentScenario - The Executable
Module A0000000041010 is present on the SE in the ELF A000000004
INFO 2012-04-05 09:31:53,812 31718ms GenerateDeploymentScenario - The instance
A0000000041010 is already installed
INFO 2012-04-05 09:31:53,812 31718ms GenerateDeploymentScenario - The instance
A000000004101001 of the Executable Module A0000000041010 will be installed
from the elf A000000004
```

Envoi du script de déploiement au SE

```
INFO 2012-04-05 09:31:54,578 32484ms notifyTransmit - >> 00 A4 04 00 08 A0 00 00
00 03 00 00 00 00
INFO 2012-04-05 09:31:54,578 32484ms notifyTransmit - << 6F 10 84 08 A0 00 00 00
03 00 00 00 A5 04 9F 65 01 FF 90-00

[Initialisation du canal sécurisé]
INFO 2012-04-05 09:31:54,640 32546ms notifyTransmit - >> 80 50 04 00 08 11 22 33
44 55 66 77 88 00
INFO 2012-04-05 09:31:54,640 32546ms notifyTransmit - << 00 00 10 23 8F 77 1D 80
21 17 04 02 00 A4 2D AA 4E AB 20 3C D5 2D DB 5A FD 5D 00 16 90-00
```



```

INFO 2012-04-05 09:31:54,671 32578ms notifyTransmit - >> 84 82 01 00 10 E7 7F 99
      B0 39 47 AF 59 A1 F1 AC D0 9A 41 9A 6A 00
INFO 2012-04-05 09:31:54,687 32593ms notifyTransmit - << 90-00

[INSTALL for install de la nouvelle instance A0 00 00 00 04 10 10 01]
INFO 2012-04-05 09:31:55,203 33109ms notifyTransmit - >> 84 E6 0C 00 25 05 A0 00
      00 00 04 07 A0 00 00 00 04 10 10 08 A0 00 00 00 04 10 10 01 01 00 02 C9 00 00
      AC 9C 67 67 AD E6 3F 84 00
INFO 2012-04-05 09:31:55,203 33109ms notifyTransmit - << 00 90-00

-----
Personalisation de l'instance
-----
INFO 2012-04-05 09:31:55,453 33359ms notifyTransmit - >> 00 A4 04 00 08 A0 00 00
      00 04 10 10 01 00
INFO 2012-04-05 09:31:55,453 33359ms notifyTransmit - << 6F 18 84 08 A0 00 00 00
      04 10 10 01 A5 0C 50 0A 4D 61 73 74 65 72 43 61 72 64 90-00
INFO 2012-04-05 09:31:55,734 33640ms notifyTransmit - >> 84 E2 A0 00 B3 01 01 7F
      9F 6C 02 00 01 56 3E 42 35 34 31 33 31 32 33 34 35 36 37 38 34 38 30 30 5E 53
      55 50 50 4C 49 45 44 2F 4E 4F 54 5E 30 39 30 36 31 30 31 33 33 30 30 30 33 33
      33 30 30 30 32 32 32 32 32 30 30 30 31 31 31 31 30 9F 64 01 03 9F 62 06 00 00
      00 38 00 00 9F 63 06 00 00 00 00 E0 E0 9F 65 02 00 0E 9F 66 02 0E 70 9F 6B 13
      54 13 12 34 56 78 48 00 D0 90 61 01 90 00 99 00 00 00 0F 9F 67 01 03 A0 01 0B
      00 00 40 00 00 00 00 77 80 99 D3 A0 02 10 52 29 A2 B1 82 0F 32 13 CA F2 24 3C
      B1 9C 5D F7 DE 65 E2 9F 48 C7 F2 12 67 CE 45 B3 79 E8 E9 80
INFO 2012-04-05 09:31:55,734 33640ms notifyTransmit - << 90-00

```

Listing A.4: Récupération des données de configuration depuis l'applet chargement dans le framework et requête identique de déploiement

Annexe B

Ontologie OWL GlobalPlatform

B.1 L'ontologie GlobalPlatform

L'ontologie GlobalPlatform est disponible à l'adresse <http://www.ecole.ensicaen.fr/~vincentalimi/GlobalPlatform.owl>

B.2 Représentation graphique de l'ontologie Global-Platform

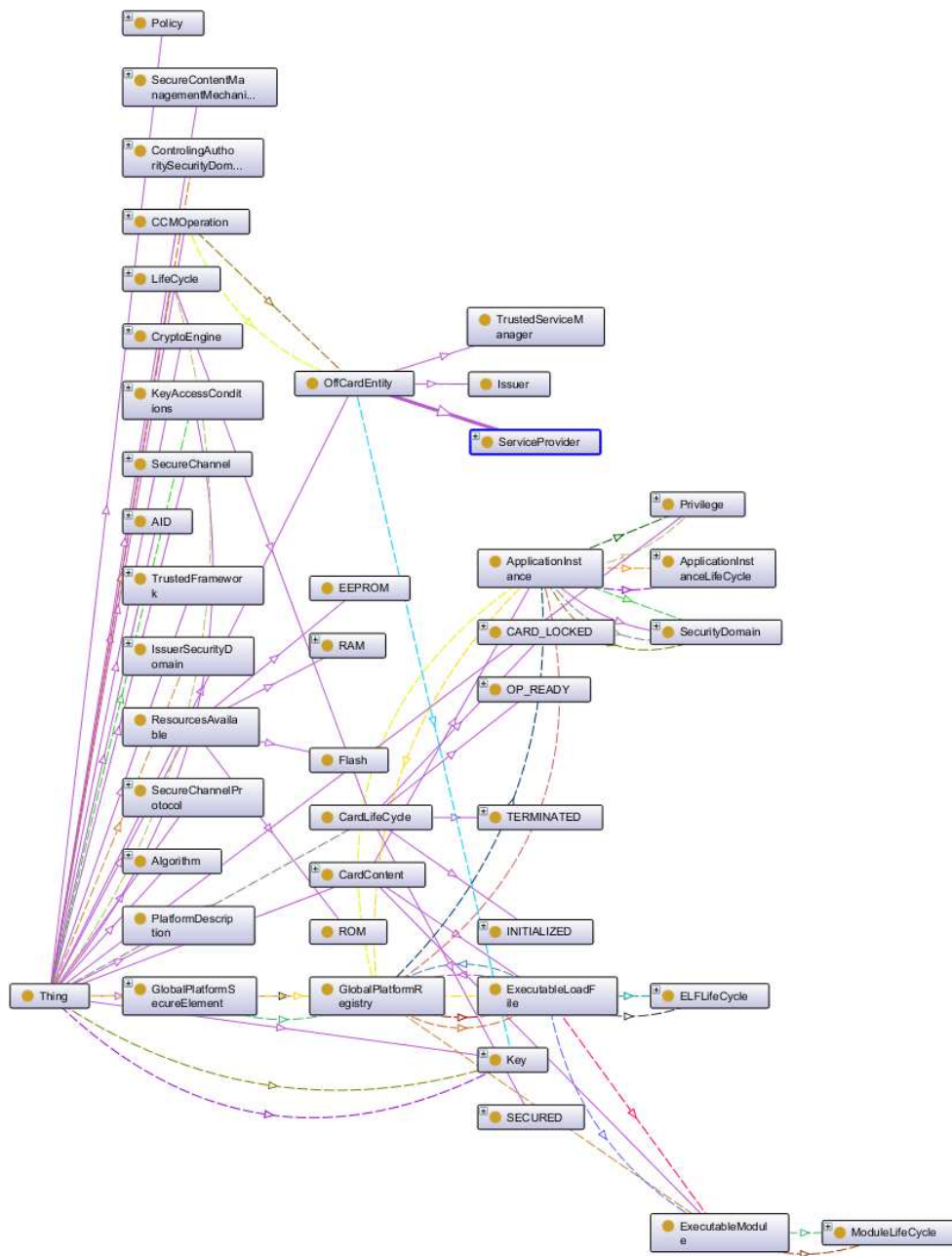


FIGURE B.1 – Représentation sous forme d’arbre horizontal des principaux concepts de l’ontologie GlobalPlatform

Annexe C

Implémentation de référence

C.1 Implémentation de référence de l'application MasterCard Paypass Magstripe

Le code source de l'implémentation de référence est accessible à l'adresse <http://www.ecole.ensicaen.fr/~vincentalimi/PaypassMagstripeSimulation.cs>

Liste des abréviations

<i>AID</i>	Application IDentifier
<i>APDU</i>	Application Protocol Data Unit
<i>API</i>	Application Programming Interface
<i>CCM</i>	Card Content Management
<i>CDA</i>	Combined DDA/Generate Application Cryptogram
<i>CMS</i>	Card Management System
<i>CVM</i>	Cardholder Verification Method
<i>DDA</i>	Dynamic Data Authentication
<i>EMV</i>	Eurocard MasterCard Visa
<i>GP</i>	GlobalPlatform
<i>IAC</i>	Issuer Action Code
<i>ISD</i>	Issuer Security Domain
<i>NFC</i>	Near Field Communication
<i>OTA</i>	Over The Air
<i>OWL</i>	Web Ontology Language
<i>PA – DSS</i>	Payment Application - Data Security Standard
<i>PCI</i>	Payment Card Industry
<i>PCIDSS</i>	PCI Data Security Standard
<i>PIN</i>	Personal Identification Number
<i>RDF</i>	Resource Description Framework
<i>RFID</i>	Radio Frequency IDentification
<i>SAM</i>	Secure Access Module
<i>SDA</i>	Static Data Authentication
<i>SE</i>	Secure Element
<i>SIM</i>	Subscriber Identity Module
<i>TAC</i>	Terminal Action Code
<i>TEE</i>	Trusted Execution Environment
<i>TES</i>	Transaction Electroniques Sécurisées

<i>TPE</i>	Terminal de Paiement Electronique
<i>TSM</i>	Trusted Service Manager
<i>TVR</i>	Terminal Veirification Results
<i>UICC</i>	Universal Integrated CIrcuit Card
<i>UML</i>	Unified Modeling Language

Table des figures

2.1	Le périmètre de la chaîne d'une transaction électronique sécurisée (source : Pôle Transactions Électroniques Sécurisées [19])	10
2.2	Pile de protocole Bluetooth®	13
2.3	Couplage par induction	16
2.4	Transfert de données par modulation autour d'une fréquence porteuse de 13,56 MHz (ISO 14443)	16
2.5	Modes de fonctionnement du NFC (source : INSIDE Secure)	17
2.6	Architecture fonctionnelle de la technologie NFC intégrée dans un mobile (source : INSIDE Secure)	18
2.7	Architecture technique du NFC (source : ENSICAEN)	19
2.8	Architecture de Java Card™ (source : Addison-Wesley)	24
2.9	Architecture de MULTOS™ (source : Damien Sauveron)	25
2.10	Architecture d'une carte GlobalPlatform (source : GlobalPlatform)	27
2.11	Architecture du TEE en cours de standardisation à GlobalPlatform (source : Trusted Logic)	29
2.12	Représentation de l'écosystème NFC (source : INSIDE Secure)	32
3.1	Cycle de vie d'un service mobile sans contact	37
3.2	Gestion en mode <i>Simple</i> , exemple de l'UICC (source : GlobalPlatform)	37
3.3	Gestion en mode <i>Dual Management</i> , exemple de l'UICC (source : GlobalPlatform)	38
3.4	Structure des données de reconnaissance de la carte (Source : GlobalPlatform)	41
3.5	Cinématique de la distribution des clés à la demande	43
3.6	Modèle d'IFM et de terminal (source : EMVCo)	50
3.7	Exemples de lecteurs et terminaux sans contact	52

3.8	Les applications de paiement et le SAM dans un module externe	54
3.9	Les applications de paiement dans le processeur d'applications et la cryptographie dans le Secure Element	55
3.10	Les applications de paiement et la cryptographie dans le Secure Element	56
3.11	Éléments mis en œuvre dans la solution proposée	62
3.12	Framework de modélisation d'un SE	71
3.13	L'interface utilisateur de l'outil Protégé	74
3.14	L'ontologie GlobalPlatform	77
3.15	Cinématique de déploiement d'un service mobile sans contact	84
3.16	Banc d'évaluation utilisé pour la validation	86
3.17	Architecture et cinématique d'utilisation du TSM Lab	88
3.18	Site web du TSM Lab	90
3.19	Étapes de la validation sur un équipement réel : (a) « réveil » du Wallet, (b) réception du script de déploiement, (c) instantiation du service mobile et de la carte virtuelle correspondante	91
4.1	Cinématique d'une transaction EMV	97
4.2	Static Data Authentication (Source : INSIDE Secure)	101
4.3	Dynamic Data Authentication (Source : INSIDE Secure)	102
4.4	Attaque par relais sur un mobile NFC	105
4.5	Matériel utilisé pour mener l'attaque (source: [136] courtoisie de Murdoch <i>et al.</i>)	106
4.6	Architecture générale de l'API	112
4.7	Diagrammes de classe partiels du module "Core"	113
4.8	Diagramme de classe partiels du module "Stack"	114
4.9	Fenêtre principale de l'interface graphique de <i>WinSCard Tools</i>	114
4.10	Architecture de l'interface graphique	115
4.11	Illustration des modes de fonctionnement de la couche Interactive	116
4.12	Interface graphique du plugin ISO 7816	117
4.13	Implémentation logicielle de l'attaque sur le code PIN	118
4.14	Le plugin "EMV Explorer"	119
4.15	Dispositif d'attaque	121
4.16	Modification du type de cryptogramme retourné par la carte	122
4.17	Modèle du protocole EMV selon Lancia dans [157]	127
4.18	Exemples de descriptions de tags contenus dans le dictionnaire	130
4.19	Croisement de deux individus	133
4.20	Données du CDOL 1 de l'application MasterCard M/Chip	134

4.21	Données du <code>Card Verification Results</code> de l'application MasterCard M/Chip utilisés pour le critère d'évaluation	135
4.22	Utilisation du simulateur JCOP pour réaliser les sessions de fuzzing	136
4.23	Matrice d'acceptation de l'application MasterCard Paypass Magstripe (source : [173])	139
4.24	Architecture du framework de fuzzing basée sur une implémentation de référence	140
4.25	Cinématique d'une transaction MasterCard Paypass Magstripe	141
4.26	Diagramme UML du framework de fuzzing	144
4.27	Mise en œuvre de l'implémentation de référence : personnalisation et construction de la matrice d'acceptation des commandes	149
4.28	Mise en évidence par l'implémentation de référence d'une non conformité dans l'application cible	150
4.29	Matrice des états de destination de la commande <code>GET PROCESSING OPTIONS</code>	150
4.30	Acceptation de la <code>GET DATA</code> pendant la phase d'utilisation	151
4.31	Acceptation de la commande <code>READ RECORD</code> pour l'enregistrement '01' dans le fichier '0A'	152
4.32	Acceptation de la commande <code>READ RECORD</code> pour l'enregistrement '01' dans le fichier '0B'	152
B.1	Représentation sous forme d'arbre horizontal des principaux concepts de l'ontologie GlobalPlatform	188

Liste des tableaux

3.1	Comparaison des solutions proposées	42
3.2	Comparaison des solutions proposées	59
3.3	Comparaison des modèles étudiés selon les propriétés attendues	70
3.4	Comparaison des temps d'exécution des différentes phases	87
4.1	Structure d'une C-APDU	128
4.2	Structure d'une R-APDU	128
4.3	Synthèse des éléments et paramètres de l'expérimentation	132
4.4	Synthèse des éléments et paramètres de l'expérimentation	142

Les transactions électroniques ont ouvert la voie à une multitude de possibilités déclinées sous diverses formes : le portail internet d'une banque pour consulter ses comptes, une carte à puce permettant d'ouvrir une porte ou de valider son titre de transport en commun, une application téléchargée sur un ordinateur ou un équipement mobile comme un assistant personnel numérique ou un téléphone portable. Cette dernière catégorie d'équipement mobile est extrêmement porteuse en terme d'offres de services. En effet, un mobile est un équipement nomade, connecté à l'Internet avec des débits de plus en plus élevés, et il est aussi de plus en plus puissant. Avec l'avènement de la technologie NFC (*Near Field Communication*) et des architectures sécurisées, le mobile a maintenant la possibilité d'héberger des applications sensibles dans une puce sécurisée, appelée « Secure Element ». Contrairement aux facteurs de formes plastiques, le Secure Element est un environnement dynamique sur lequel on peut charger des applications qui seront accessibles par un lecteur comme un terminal de paiement ou un lecteur de badge d'accès à un bâtiment. On parle alors de *service mobile sans-contact* ou de *service mobile NFC*.

Nous proposons dans cette thèse d'apporter plusieurs contributions pour faciliter le déploiement de services mobiles exploitant un Secure Element. L'adoption de ce type de service pour des applications à forte valeur ajoutée passe par une infrastructure et des outils partagés par les différents acteurs. Dans un premier temps, nous proposons trois contributions pour l'aide au déploiement de services mobiles basé sur un Secure Element afin de : faciliter la personnalisation d'un Secure Element par un TSM non propriétaire de celui-ci, faciliter l'échange de clés pour le TSM et réaliser une transaction avec un téléphone mobile comme Terminal de Paiement Électronique). Dans un second temps, nous nous intéressons à l'analyse de la sécurité des transactions de paiement.

Contributions to the deployment of mobile services and to the analysis of transactions security

Electronic transactions have paved the way for a multitude of services in various forms : a home banking web portal, an access control smart card opening a door or paying a transit fare, or an application downloaded onto a computer or a mobile device such a personal digital assistant or a mobile phone. The latter category of mobile equipment is extremely promising in terms of service offers. Indeed, a mobile handheld device is connected to the Internet with speeds higher and higher and is also more powerful. With the advent of the NFC (*Near Field Communication*) technology and secure architectures, mobile devices now has the ability to host sensitive applications in a secure chip, called « Secure Element ». Unlike plastic form factors, the Secure Element is a dynamic environment where you can download applications that will be accessible by a reader such as a point of sale or an access control reader. Those services are called *mobile contactless services* or *NFC mobile services*.

We propose in this thesis several contributions to facilitate the deployment of mobile services based on a Secure Element. The adoption of this high added value services relies on an infrastructure and tools shared by different actors. As a first step, we propose three contributions to aid the deployment of mobile services Secure Element based allowing to : facilitate the personalization of a Secure Element by a TSM non-owner thereof, facilitate the exchange of keys for the TSM and perform a transaction with a mobile phone as a point of sale. In a second step, we focus on the analysis of payment transactions.

Indexation Rameau : MOTS CLÉS À VOIR AVEC LA BU

Indexation libre : Monétique - Paiement mobile - Services mobiles - Aide au déploiement - Analyse de la sécurité des transactions

Informatique et applications

Laboratoire GREYC - UMR CNRS 6072 - Université de Caen Basse-Normandie - Ensicaen
6 Boulevard du Maréchal Juin - 14050 CAEN CEDEX