Une architecture d'administration de cartes à puce, similaire à OTA, et dédiée aux réseaux sans fil IP

Toundé Mesmin Dandjinou

TELECOM PARIS
école nationale supérieure des télécommunications

Travaux de recherche pour l'obtention de la

Thèse de doctorat

de l'Ecole Nationale Supérieure des Télécommunications de Paris

Spécialité : Informatique et Réseaux

Par

Mesmin Toundé DANDJINOU

Une architecture d'administration de cartes à puce, similaire à OTA, et dédiée aux réseaux sans fil IP

Soutenue le 19 juin 2006 devant le jury composé de

Président :	Pr Pierre PARADINAS, *CNAM, Paris, France*
Rapporteurs :	Pr Bernard COUSIN, *Université de Rennes 1, Rennes, France*
	Pr Pascal LORENZ, *Université de Haute Alsace, Colmar, France*
Directeur de thèse :	Pr Pascal URIEN, *ENST-Paris, Paris, France*
Examinateurs :	Pr Guy PUJOLLE, *Université de Paris VI, Paris, France*
	M. Eric ALZAI, *Oberthur Card System, Nanterre, France*

Gloire à DIEU !
Gloire à DIEU !
Gloire à DIEU !
A mon épouse Sandrine,
A mes enfants Carmen, Jean-Emmanuel, Anne et Marie-Esther,
A mes parents.

Remerciements

Au moment d'écrire ces lignes de remerciements, je suis dans la reconnaissance vis-à-vis de mon Dieu qui m'a témoigné de son amour et de sa fidélité pendant toutes ces années. Il fallait vraiment du courage pour se replonger dans une aventure de thèse, et je pense qu'en Lui je n'en ai pas manqué. Merci infiniment à Lui pour toutes ses grâces et ses bénédictions.

Elles sont nombreuses les personnes qui se sont battues pour que je puisse en son temps trouver une formation de DEA, un laboratoire d'accueil, une bourse, les documents de ma mise en route, un endroit où loger, travailler, etc. J'ai été aidé, aussi bien avant que durant mes multiples séjours de recherches en France. J'ai été soutenu aussi bien par des personnes morales que physiques, burkinabé comme françaises ; qu'elles trouvent ici exprimés mes sincères remerciements.

Je dis ici toute ma particulière reconnaissance à la coopération française qui a pris en charge le financement de cette thèse en alternance.

Grand merci au professeur Guy Pujolle qui a accepté mon dossier d'inscription pour le DEA et qui m'a fait l'honneur de faire partie de mon jury de thèse. Il m'a fait confiance dès le départ et cela m'a encouragé à redoubler d'efforts, moi qui avais au départ pas mal de handicaps à franchir.

Mon patron, le professeur Pascal Urien, malgré sa surcharge de travail a manifesté une très grande disponibilité à mon endroit tout au long de ce travail. Il m'a communiqué sa passion pour les cartes à puce. Et si j'ai pu avancer, c'est beaucoup grâce à lui. Je lui dis de tout cœur grand merci.

Merci aux professeurs Bernard Cousin et Pascal Lorenz qui ont accepté être les rapporteurs de cette thèse. Merci pour les critiques pertinentes qui ont été formulées et qui m'ont servi à améliorer la qualité de la présentation de mes travaux.

Je suis reconnaissant au professeur Pierre Paradinas qui a accepté de présider mon jury de soutenance de thèse.

Merci à M. Eric Alzai qui a consenti à faire partie de mon jury malgré son emploi du temps très chargé.

J'exprime ma gratitude aux collègues de bureau et de laboratoire en général qui, de diverses manières, ont, par leur présence à mes côtés, contribué à me faire passer les soucis de tout bon père de famille en mission de travail loin des siens.

Merci à tous ces amis et parents habitant Paris et sa banlieue, Annemasse, Besançon, Bobo-Dioulasso, Crissey, Dijon, Le Mans, Nice, Ouagadougou, Poitiers, Strasbourg et

Tours, qui m'ont supporté durant la réalisation de cette deuxième thèse. En particulier à tous ceux qui, ce 19 juin 2006, ont, soit eu des pensées pour moi, soit fait le déplacement pour assister à ma soutenance, je dis ici toute ma gratitude.

Je suis très reconnaissant à Geneviève Jomier et Anne-Marie Charles qui, non comptant des mètres-cubes à Paris qu'elles ont continuellement et gracieusement mis à ma disposition, n'ont ménagé aucun effort pour que mon travail puisse se dérouler et être présenté dans les meilleures conditions possibles. Soyez en retour richement bénies !

Merci aux frères et sœurs de sang comme d'église à Bobo, Ouagadougou et Paris dont le soutien constant et multiforme ne m'a pas fait défaut durant toutes ces années de va-et-vient entre le Burkina Faso et la France.

Merci à tous !

Table des matières

Liste des figures

Liste des tableaux

Abstract

IP wireless networks are invading most of our life areas. But the lack of secured access of these networks is a serious brake for the development of new services in them. In our work, we propose to use smart cards as security modules, as it is the case in the mobile radio telephony networks GSM.

For this purpose in spite of smart cards limitations of their computational and storage capabilities, we suggest a new protocol named EAP-SSC (EAP Secured Smartcard Channel). This protocol is dedicated to the mutual authentication using both symmetrical and asymmetrical cryptographic keys contexts.

As IP wireless networks are operated by various administrative authorities, it is necessary to anticipate the consideration of the diversity of the underlined security politics. So, we propose a platform named OpenEAPSmartcard intended to be set up in every Java card of the market place. This platform is opened and easy for being convenient for new authentication scenarios chosen by the computer programmers.

Security of cryptographic materials stored on the servers is not safeguarded, because of the operating systems flaws and vulnerabilities; access points closed to the users are less sheltered from attacks. For that reason we suggest to create authentication micro-servers that correspond to EAP servers included in the Java smart cards.

The deployment of those micro-servers on a large scale will cause the problem of keeping them up to date. We propose a software architecture called TEAPM (Trusted EAP Module) which heart is formed by EAP and EAP-TLS surrounded by XML and HTTP protocols. This architecture allows the "On The Air" secured administration of the micro-servers

Key-words : GSM, EAP, smart cards, TLS, EAP-SSC, OpenEAPSmartcard, authentication micro server, OTA, TEAPM, tracks servers.

Résumé

Les réseaux sans fil IP envahissent nos lieux de vie et le défaut de sécurité d'accès est un sérieux frein au développement de nouveaux services en leur sein. Dans ce travail nous proposons l'emploi de la carte à puce Java comme module de sécurité pour l'accès à ces réseaux, comme le sont les puces pour la téléphonie mobile GSM.

Pour y parvenir malgré les limitations de ces cartes en matière de puissance de traitement et de capacité de stockage, on propose un nouveau protocole du nom de EAP-SSC (EAP Secured Smartcard Channel). Il assure une authentification mutuelle fondée sur la cryptographie à clés symétriques ou asymétriques.

La diversité des autorités administrant les réseaux sans fil IP commande la prise en compte d'une variété de politiques de sécurité applicables. Aussi, proposons-nous une plate-forme dénommée OpenEAPSmartcard pour toute carte Java du marché. Son architecture est ouverte et facile à adapter aux scénarii d'authentification des développeurs.

La sécurité des matériaux cryptographiques stockés sur les serveurs n'est pas garantie, à cause des attaques profitant des failles et des vulnérabilités des systèmes d'exploitation ; celle des bornes d'accès à la portée des utilisateurs l'est moins encore. Notre solution est d'implanter dans les cartes Java des serveurs EAP dénommés micro-serveurs d'authentification.

Le déploiement de ces micro-serveurs pose le problème de leur mise à jour dans le temps et dans l'espace. Une architecture logicielle dénommée TEAPM (Trusted EAP Module) est proposée. En son cœur sont les protocoles EAP et EAP-TLS surmontés de XML et HTTP pour faciliter l'administration distante et sécurisée "Over The Air" des cartes à puce Java.

Mots-clés : GSM, EAP, carte à puce, TLS, EAP-SSC, OpenEAPSmartcard, micro-serveur d'authentification, OTA, TEAPM, serveurs de traces.

General introduction

Nowadays two families of networks can be distinguished : operator wireless networks in which the radio telephony GSM (*Global System for Mobile communications*) appeared during ninety years is a part, and the IP (*Internet Protocol*) wireless networks well known under the WLAN IP (*Wireless Local Area Network Internet Protocol*) name which ancestor is the IEEE 802.11 standard appeared by ninety nine years.

Operator networks are closed because the acceptation of an user request is conditioned by subscribing to one of the network operators. They are also closed when we consider the new services which can be installed only with the agreement of the operator. Generally, networks of this family operate well, because of the strict control of user accesses (identification and authentication of the subscriber), and one must pay for several offered services which require authentication and authorization.

In the opposite side, IP wireless networks are equivalent to an opened world. Many networks of this kind exist in our towns and any user can connect to them, because their default configuration does not realize an access control. Few services are offered to the users in these networks. However, the capacities in terms of bandwidth of these IP wireless networks are increasing and some of them compete with and sometimes surpass some classical networks. For example, with the IEEE 802.11b a maximum throughput of 11 Mbit/s is expected, that is taller than the classical Ethernet wired networks 802.3 capacity. With the IEEE 802.11g standard offering a theoretical throughput of 54 Mbit/s, we obtain on the air support almost the half of the capacity of the Fast Ethernet wired networks. Furthermore, since 2005 the IEEE 802.16e specifications plan to offer on the radio more than 100 Mbit/s for subscriber stations moving at vehicular speeds and combining fixed and mobile broadband wireless access. This opportunity can sound the death knell of the UMTS (*Universal Mobile Telecommunications System*) technology.

For a real development of the IP wireless networks and their massive usage, it is important to set prior in these networks security services that allow the controlled networks to be launched at their actual position. The main issue to solve in this work will be to propose an ideal framework for the development of services in the IP wireless networks.

The first brick to use in the assembling of this suitable framework for services development in IP wireless networks is the security of the user access to these networks. In GSM networks the SIM (*Subscriber Identity Module*) card is used to identify and authenticate the subscriber of a mobile phone network. We think a similar system must be introduced to ensure the security of the user access to IP wireless networks.

Basically, the IP wireless networks are from the management point of view under several administrative authorities, a situation that is not possible to observe in an operator network. For example the IP wireless networks at home are under the administrative authority of natural persons. In the companies and public areas like hot spots, these networks are under the administrative responsibility of a legal entity. Each network has its particular security policy. For maintaining this administration liberty in spite of the strengthening of the security of users access to the IP wireless networks, the security module has to support a wide range of authentication scenarios easy to implement. The

second contribution of our work is to propose for this module an opened platform, adaptable and not linked to a particular manufacturer.

Although the physical access to the different equipments in the operator core network is highly secured, some of them can become compromised, after attacks using the flaws and vulnerabilities of operating systems they run. In a similar situation of compromised machines, it is no more possible to ensure confidentiality and integrity of cryptographic materials they store. In the IP wireless networks, machines like servers and access points can be in a physical scope of users : their security must be increased. How ? Through our third contribution, we try to give an answer to this issue.

Finally, considering the great number of networks a user will be able to access, it is essential for the module which contains profiles, identity and authorization credentials of the user, to be remotely manageable. In GSM networks, the OTA (*Over the Air*) administration allows the operator a remote control of the configuration of the security modules related to the users. In the same way, the means to download code and data in the security modules have to be anticipated in IP wireless networks. The proposition of an architecture capable to allow this remote administration will be the last contribution of this work.

The presentation of this work is structured in two main parts. In the beginning part we realize a state of the art concerning on the one hand security in GSM and IP wireless networks, and on the second hand EAP (*Extensible Authentication Protocol*) smart card. In the final part we show our contributions to improve security in IP wireless networks and so promote the development of services inside these networks.

Chapter 1 starts with the presentation of the concept of smart cards administration using OTA technology and the general framework of its deployment, the GSM network. A description of transport services on which this technology is backing on is also done.

Chapter 2 concerns interactions between functional parts of the mobile phone, particularly SIM card and mobile equipment. We describe mechanisms used to ensure first coherence during the realization of basic and standardized supplementary services, and secondly open towards all new supplementary services which certainly will be set up to respond to the customers needs.

In the chapter 3 are at first described 802.11 specifications with the reinforcement of the access security thanks to 802.1X infrastructure. Then, the presentation of 802.11i architecture shows the manner to ensure nowadays the security of IP wireless local networks by using required ciphering and keys distribution protocols. Emerging 802.16 and 802.16e metropolitan IP wireless networks are not forgotten since they are expected to offer Internet broadband everywhere. As a numerous group of key materials to manufacture and distribute are needed to provide security in these networks, the chapter is ended by noticing that a SAM (*Secure Access Module*) card would be convenient to IP wireless networks.

The first part of the manuscript is ended with chapter 4 in which EAP (*Extensible Authentication Protocol*) and some strong authentication methods usable in wireless network contexts are presented. Then, the concretisation of this protocol inside a Java card is described and we show the main constraints to take account if we want to use it to reinforce the security of IP wireless networks accesses.

Our first contribution is presented in the chapter 5. Considering the limited storage and computation capacities of the smart cards, we proposed to IETF (*Internet Engineering Task Force*) the draft of the protocol named EAP-SSC (*EAP Secured Smartcard Channel*). This

protocol is in charge of mutual authentication between the subscriber and the visited IP wireless network, in both symmetrical and asymmetrical cryptographic keys contexts.

In chapter 6 is presented our second contribution : the opened *OpenEapSmartcard* platform intended to compute EAP inside a Java card. Its purpose is to facilitate the deployment of a variety of solutions which will make more secure the access of IP wireless networks, by offering a software architecture that can be supported by the common smart cards on the market and easily reusable. The difficulties we meet during the realization of this platform are pointed out. Some performance test results of the implementation of this platform are given, using smart cards made by different manufacturers, and using also several authentication methods.

Chapter 7 brings out the concept of EAP servers implemented in Java cards, our third contribution. Actually, in addition to the first mission of storing confidential documents related to the user, it is necessary to foresee in the IP wireless network environments a collector agent of reliable and contextual data which can be used in the future to produce any kind of piece of proof. The EAP server presence inside the smart card, able to initiate EAP requests towards EAP clients, can help to assure this service.

Finally, we propose in the chapter 8 a software machine which goal is to promote *Over The Air* administration of smart cards inside IP wireless networks. The heart of this device is composed of EAP and EAP-TLS *(EAP Tranport Layer Security*) protocols surrounded by XML (*eXtensible Markup Language*) and HTTP (*Hypertext Transfer Protocol*) which open the smart card to the distributed applications of Web : the TEAPM (*Trusted EAP Module*). We show a few cases of utilization of this device and deliver some experimental results.

Introduction générale

Deux familles de réseaux sans fil se distinguent de nos jours : les réseaux sans fil d'opérateurs dont fait partie celui de la radio téléphonie GSM (*Global System for Mobile communications*) apparue dans les années 90, et les réseaux sans fil IP (*Internet Protocol*) connus sous l'appellation WLAN IP (*Wireless Local Area Network Internet Protocol*) dont le standard IEEE 802.11 est l'ancêtre paru dans les années 1999.

Les réseaux sans fil d'opérateurs sont fermés en ce sens qu'un utilisateur ne peut y être admis que s'il a auparavant souscrit un abonnement chez un opérateur. Ils sont également fermés par rapport aux nouveaux services qui ne peuvent y être introduits que sous le contrôle d'un opérateur. Les réseaux de cette famille fonctionnent en général bien, car l'accès des utilisateurs à ce type de réseaux est strictement contrôlé (identification et authentification de l'abonné). De nombreux services payants, qui nécessitent une authentification et une autorisation, y sont offerts.

A l'opposé, les réseaux sans fil IP correspondent à un monde ouvert. En effet, de nombreux réseaux de ce type existent dans nos villes et n'importe quel utilisateur peut s'y connecter, leur configuration par défaut ne réalisant aucun contrôle d'accès. Très peu de services y sont offerts à l'utilisateur. Or les capacités en termes de bande passante de ces réseaux sans fil IP sont en train de rivaliser et même de dépasser celles de certains réseaux classiques. Par exemple avec le standard IEEE 802.11b un débit théorique maximum de 11 Mbit/s est prévu, ce qui dépasse les capacités des réseaux Ethernet filaires 802.3 classiques. Avec le standard IEEE 802.11g offrant un débit théorique de 54 Mbit/s, on a sur le médium aérien presque la moitié des capacités offertes dans les réseaux câblés *Fast Ethernet*. De plus, depuis 2005 les spécifications IEEE 802.16e prévoient d'offrir via le médium hertzien une bande passante de plus de 100 Mbit/s aux terminaux en déplacement à l'allure d'un véhicule. Ceci pourrait sonner le glas de l'UMTS (*Universal Mobile Telecommunications System*).

Pour que les réseaux sans fil IP puissent connaître un réel développement de leur utilisation par un plus grand nombre, il est impérieux qu'y soient déployés les services de sécurité qui ont permis aux réseaux contrôlés par les opérateurs d'occuper la position qu'ils ont actuellement. Le problème à résoudre dans notre travail est donc de proposer un cadre favorable au développement des services dans les réseaux sans fil IP.

La première brique à employer dans le montage de ce cadre propice au développement des services dans les réseaux sans fil IP est la sécurité de l'accès des utilisateurs de ces réseaux. Dans le réseau GSM la carte SIM (*Subscriber Identity Module*) sert de module d'identification et d'authentification de l'abonné au réseau téléphonique mobile. Nous pensons qu'un dispositif analogue doit être introduit pour sécuriser l'accès de l'utilisateur aux réseaux sans fil IP.

Fondamentalement les réseaux sans fil IP sont des réseaux qui, du point de vue de leur administration, sont sous différentes autorités, contrairement à ce qu'on rencontre dans un réseau d'opérateur. Par exemple les réseaux sans fil IP dans les domiciles sont sous l'autorité administrative de personnes physiques. Dans les entreprises comme dans les lieux publics (*hot spots*), ils sont sous la responsabilité administrative d'une personne morale. Chacun de ces réseaux a sa propre politique de sécurité. Pour que cette liberté

d'administration puisse perdurer malgré le renforcement de la sécurité de l'accès des utilisateurs aux réseaux sans fil IP, le module de sécurité devra être en mesure de supporter une diversité de scenarii d'authentification faciles à implémenter. Proposer pour ce module sécurisé une plate-forme ouverte, adaptable et indépendante du fabricant de carte à puce constituera la deuxième contribution de notre travail.

Bien que l'accès physique aux différents équipements situés au cœur des réseaux d'opérateurs soit hautement sécurisé, il arrive néanmoins que certains d'entre eux soient compromis, suite à des attaques profitant des failles et vulnérabilités des systèmes d'exploitation qu'ils hébergent. En situation de compromission de ces machines, on ne peut pas garantir la confidentialité et l'intégrité des matériels cryptographiques qu'elles conservent. Dans les réseaux sans fil IP, les équipements tels que les serveurs et les points d'accès peuvent être à la portée physique des utilisateurs : leur sécurité devrait donc être renforcée. Comment ? A travers notre troisième contribution nous tenterons de répondre à cette interrogation.

Enfin, vu le grand nombre de réseaux sans fil IP auxquels un utilisateur pourra se connecter, il est indispensable que le module de sécurité renfermant les profils, les pièces d'identités et autorisations de l'utilisateur, puisse être administré à distance. Dans les réseaux GSM, l'administration OTA, *Over The Air*, permet de prendre à distance le contrôle de la configuration des modules de sécurité relatifs aux utilisateurs. De même dans les réseaux sans fil IP le moyen d'y télécharger des données ou du code exécutable est à prévoir. La proposition d'une architecture en mesure de rendre possible cette administration distante sécurisée constituera notre dernière contribution dans ce travail.

L'exposé de notre travail est structuré en deux grandes parties. Dans la première partie nous faisons l'état de l'art sur la sécurité dans les réseaux GSM et sans fil IP d'une part, et sur la carte à puce EAP d'autre part. Dans la deuxième partie nous exposons nos contributions qui devraient améliorer la sécurité des réseaux sans fil IP et y favoriser le développement de services.

Le chapitre 1 débute avec la présentation du concept d'administration des cartes à puce via la technologie OTA et du cadre général de sa mise en œuvre, le réseau GSM, avec une description des services de transport en dehors de celui de la voix sur lequel cette technologie s'appuie.

Le chapitre 2 traite des interactions entre les entités fonctionnelles constituant le téléphone mobile lui-même, en particulier la carte SIM et l'équipement mobile. Sont décrits les mécanismes employés pour assurer d'une part une cohérence dans la réalisation des services de base et des services supplémentaires standardisés, et d'autre part une ouverture vis-à-vis de nouveaux services supplémentaires qui, immanquablement se mettront en place en vue de répondre aux besoins des utilisateurs.

Au chapitre 3 sont d'abord décrites les spécifications 802.11 avec le renforcement de la sécurité d'accès à travers l'infrastructure 802.1X. Ensuite la présentation de l'architecture 802.11i montre comment est actuellement garantie la sécurité dans les réseaux locaux sans fil IP en matière de protocoles requis pour le chiffrement et la distribution des clés. Les réseaux sans fil IP métropolitains 802.16 et 802.16e émergeants ne sont pas oubliés puisqu'ils sont sensés offrir des accès Internet haut débit partout. De très nombreux matériaux de fabrication et de distribution de clés étant nécessaires pour assurer la sécurité dans ces réseaux, le chapitre est conclus par le constat qu'une carte SAM (*Secure Access Module*) pourrait convenir aux réseaux sans fil IP.

La première partie du manuscrit se termine avec le chapitre 4 dans lequel sont présentés d'abord le protocole EAP (*Extensible Authentication Protocol*) et certaines

méthodes d'authentification robustes utilisables dans le contexte des réseaux sans fil. Ensuite est décrite la première implémentation de ce protocole au sein d'une carte à puce Java et sont indiquées les principales contraintes à satisfaire au cas où la carte à puce EAP devrait servir à renforcer la sécurité de l'accès aux réseaux sans fil IP.

Notre première contribution est présentée au sein du chapitre 5. Pour tenir compte des capacités limitées de stockage et de traitement de la carte à puce, nous avons proposé à l'IETF (*Internet Engineering Task Force*) le *draft* du protocole EAP-on (*EAP Secured Smartcard Channel*). Celui-ci est chargé, aussi bien dans un contexte cryptographique de clés symétriques qu'asymétriques, d'assurer une authentification mutuelle entre l'abonné et le réseau sans fil IP visité.

Le chapitre 6 présente notre deuxième contribution : la plate-forme ouverte *OpenEapSmartcard* réalisant EAP au sein d'une carte Java. Son objectif est de faciliter le déploiement d'une diversité de solutions sécurisant l'accès aux réseaux sans fil IP, en proposant une architecture logicielle supportable par les cartes Java du marché et facilement réutilisable. Nous y évoquons les difficultés apparues lors de la réalisation de la plate-forme. Nous communiquons les résultats des tests de performances de la plate-forme implantée sur des cartes de fabricants différents d'une part, et utilisant diverses méthodes d'authentification d'autre part.

Le chapitre 7 traite des serveurs EAP implémentés dans les cartes à puce Java, notre troisième contribution. En effet, en plus de cette mission première de stockage de documents confidentiels se rapportant à l'utilisateur, il est nécessaire de prévoir dans les environnements de réseaux sans fil IP un agent collecteur d'informations contextuelles fiables servant à la production de preuves de toutes sortes. La présence d'un serveur EAP sur la carte à puce, capable d'initier des requêtes EAP vers des clients EAP, peut aider à rendre ce service.

Enfin, nous proposons au chapitre 8 un dispositif logiciel visant à favoriser l'administration *Over The Air* des cartes à puce dans les réseaux sans fil IP. Le cœur du dispositif est constitué des protocoles EAP et EAP-TLS (*EAP Transport Layer Security*) au-dessus desquels XML (*eXtensible Markup Language*) et HTTP (*Hypertext Transfer Protocol*) viennent s'arrimer pour ouvrir la carte à puce aux applications Web distribuées : le TEAPM (*Trusted EAP Module*). Nous montrons des utilisations possibles de ce dispositif et présentons quelques résultats d'expérimentation.

ETAT DE L'ART

Chapitre 1 : Introduction à l'administration OTA (Over The Air)

L'utilisation de l'acronyme anglais OTA pour *Over The Air* s'est répandue durant ces dernières années. Il désigne au départ la possibilité de prendre à distance et par le biais du réseau GSM (*Global System for Mobile Communications*) le contrôle de la configuration du téléphone mobile d'un abonné [GSM01.04]. En effet, avec le développement des capacités des cartes à puce, l'application initiale de téléphonie mobile a été complétée par de nouvelles applications logées dans la même carte SIM (*Subscriber Identity Module*), afin de donner à ces nouveaux services un caractère mobile. Notre objectif dans ce chapitre est :

- de présenter d'abord le concept général de l'administration OTA,
- de décrire ensuite l'infrastructure du réseau GSM qui en supporte la mise en œuvre,
- et enfin de rappeler les deux types de transport prévus en dehors de celui de la voix, pour étendre le champ des services applicatifs mis à la disposition des clients de la radio téléphonie : le transport par messages courts et celui des données de services supplémentaires non structurées.

I. Introduction au concept de l'administration OTA

Malgré les possibilités offertes par les opérations et transactions sur l'Internet, il reste encore, dans la vie de tous les jours, de nombreuses situations nécessitant d'un client un déplacement physique vers un prestataire de services, pour un nouvel achat ou pour un dépannage. Il est vrai qu'avec la multiplication des "*hot spots*", les lieux d'accès à l'Internet se sont multipliés, mais on est encore loin d'atteindre la couverture actuellement assurée par le réseau GSM.

L'une des raisons du succès de la téléphonie mobile est la facilité avec laquelle l'abonné peut bénéficier des services offerts par ce réseau. Ainsi, quel que soit le lieu dans l'espace géographique couvert par son opérateur de radiotéléphonie et quelle que soit l'heure à laquelle l'abonné désire consulter son solde ou s'approvisionner en unités téléphoniques, il peut bénéficier de ces services. La possibilité de changer la configuration des services téléphoniques de l'abonné, à distance et sans qu'on ait besoin d'aller se brancher physiquement à un câble, va servir à l'amélioration ou à l'extension des services rendus par ces équipements électroniques qui sont actuellement les plus répandus au monde[1].

Ces remarquables possibilités de mise à disposition de nouveaux services et de modification (extension ou réduction) de celles qui existent, ne sont que la conséquence de la capacité à télécharger dans une carte à puce téléphonique SIM des informations. Ici, en plus des capacités du réseau, ce sont celles de la carte à puce employée dans le téléphone portable qui vont être sollicitées. En effet, les capacités de stockage et de traitement de la carte à puce ont connu de grandes évolutions ces dernières années. Elles permettent par exemple d'exécuter, dans des contextes bien isolés les uns des autres, les différentes applications embarquées. Les informations qui peuvent, via l'interface aérienne, aboutir à la carte SIM de l'abonné correspondent soit à des données à employer par les applications, soit à du code interprétable ou directement exécutable.

[1] En 2005, selon GSM World, on comptait plus de 1,5 milliards d'abonnés appartenant à des milliers de réseaux GSM réparties dans plus d'une centaine de pays dans le monde.

Un autre aspect de cette administration via l'interface hertzienne, est la limitation ou même l'absence d'intervention du porteur du téléphone mobile dans ce processus d'administration à distance. Pourvu que le terminal téléphonique soit en bon état de marche, allumé et dans une zone de couverture radio-téléphonique de son opérateur ou de l'un de ses partenaires[2], l'administration à distance est réalisable sans demander forcément l'implication de son porteur. Cependant, dans certaines situations, l'intervention du porteur de la carte est requise : par exemple quand l'authentification de ce dernier constitue un préalable à l'obtention de l'autorisation d'accéder aux ressources de la carte SIM. La satisfaction de cette condition ouvre la voie à de nombreux services à distance tels que le télétravail, la télésurveillance, le téléguidage, la téléconsultation, le téléchargement, le téléachat, le télépaiement, en fait tout ce qui pourrait être fait partiellement ou complètement à distance. Il est de ce fait très important que ces opérations d'administration se déroulent dans des conditions de sécurité irréprochables sur les segments carte SIM / équipement mobile d'une part, et équipement mobile / réseau d'autre part, en faisant l'hypothèse qu'au sein du réseau cœur de l'opérateur de bonnes mesures de sécurité sont déjà prises.

Cet exposé serait incomplet s'il n'évoquait pas les acteurs et les utilisateurs concernés par l'administration OTA.

Au nombre des acteurs il y a les MNO (*Mobile Network Operator*) et les MVNO (*Mobile Virtual Network Operator*). Les MNO sont les opérateurs traditionnels de radiotéléphonie ; les infrastructures déployées sur le terrain leur appartiennent. Les MVNO sont de nouveaux venus dans les télécommunications, à la faveur du dégroupage en cours dans ce secteur. Ils ne sont pas propriétaires du réseau cellulaire mais sont en mesure de proposer un service complet de téléphonie mobile. Généralement les MVNO achètent en gros des minutes de téléphonie aux opérateurs traditionnels et les revendent ensuite à leurs clients[3].

Parmi les utilisateurs de cette technologie figurent les porteurs d'équipements de téléphonie mobile et de nouveaux fournisseurs de services. A quelques exceptions près, les porteurs de terminaux de téléphonie mobile ont souscrit un abonnement auprès d'un opérateur réel ou virtuel de radiotéléphonie. Les nouveaux fournisseurs de services profitent de l'ouverture du cœur des réseaux opérateurs pour proposer d'autres services aux abonnés de la radiotéléphonie, moyennant le respect des conditions de sécurité. Qu'il s'agisse des MNO, des MVNO ou des fournisseurs de services OTA, leur administration à distance ne peut se faire que si, au cours du processus conduisant à sa personnalisation, la carte SIM a été préparée par son propriétaire pour supporter ce genre d'opération.

Comme toute l'administration réalisée à travers l'interface aérienne repose sur l'existence d'un réseau de radiotéléphonie GSM, la section suivante fournit une description générale de ce réseau.

[2] En 2005, selon la source http://www.gsmworld.com/ news/press_2006/press06_06.shtml, le partenariat GSMA (GSM Association) regroupait plus de 680 opérateurs et fabricants de téléphones mobiles dans le monde.

[3] Par exemple en 2006, la société Debitel est un MVNO partenaire de la société SFR, tandis que M6 Mobile est un MVNO partenaire de Orange.

II. Généralités sur le réseau GSM

Depuis 1992, date de sa première exploitation commerciale, le GSM sert de moyen de transmission de la voix, mais aussi des données à travers les textos[4] aussi appelés messages courts SMS (*Short Message Services*). Il fonctionne dans les bandes de fréquences de 900 MHz, 1800 MHz et 1900 MHz selon les pays, et autorise sur l'interface aérienne des débits théoriques variant entre 9,6 et 14,4 kbit/s.

Plusieurs centaines de documents de spécifications ont été produits par les différents groupes de travail du GSM. En 1989 toutes ces spécifications sont intégrées à celles de l'ETSI (*European Telecommunications Standards Institute*) qui dès lors en assure l'évolution à travers différentes phases dénommées phase 1, phase 2 et phase 2+. La phase 1 démarre en 1992 à partir de toutes les spécifications du GSM concernant l'implémentation de ses services de base tels que la transmission de la voix, le transfert d'appel, la gestion de l'itinérance (*roaming*) et le service des messages courts. La phase 2, débutée en 1996, voit la prise en compte des services supplémentaires tels que les conférences, les appels en *handover*, la négociation du numéro d'appel et la prise en compte de la bande de fréquences des 1800 MHz. Dans la phase 2+, aussi qualifiée de *2.5 G*, sont prises en compte les fonctions de la *SIM Application Toolkit*, la technologie de commutation de circuit de données à grande vitesse HSCSD (*High Speed Circuit-Switched Data*) assurant une vitesse théorique de transmission de 8 fois 9,6 kbit/s soit 76,8 kbit/s. La phase 2+ inclut aussi le GPRS (*General Packet Radio System*) qui offre un service de commutation de paquets à un débit allant théoriquement jusqu'à 171,2 kbit/s, tarifié au volume et non à la durée de connexion, et assure l'interconnexion avec d'autres réseaux paquets comme Internet et X.25. Une autre amélioration du GSM est venue avec la technologie EDGE (*Enhanced Data rate for GSM Evolution*) qui permet à un terminal mobile ayant une vitesse comprise entre 100 et 250 km/h de se connecter au réseau avec un débit de 384 kbit/s. Comme ultime évolution on a cru à la *3 G* avec l'UMTS (*Universal Mobile Telecommunications System*) [SaTh01] offrant des services multimédia à des débits compris entre 0,4 et 2 Mbit/s. Cependant, au moment de terminer cette rédaction, est présentée la technologie *3,5 G* baptisée HSDPA (*High Speed Downlink Packet Access*) supportant des vitesses de transfert des données de l'abonné mobile comprises entre 1 et 8 Mbit/s. Alors à quand la *4 G* ?

Depuis 2000 deux groupes travaillant sur le GSM existent à l'ETSI : d'une part le 3GPP (*Third Generation Partnership Project*) constitué d'experts travaillant sur l'interface de l'équipement mobile avec la SIM (*Subscriber Identity Module*) ou l'USIM (*Universal Mobile Telecommunications System Subscriber Identity Module*) ; et d'autre part le groupe EP SCP (*ETSI Project Smart Card Platform*) constitué d'experts travaillant sur tous les aspects des cartes à puce employées pour les télécommunications.

[4] En 2005 plus de 1000 milliards de textos ont été échangés dans le monde. (Source : http://www.gsmworld.com/news/press_2006/press06_06.shtml)

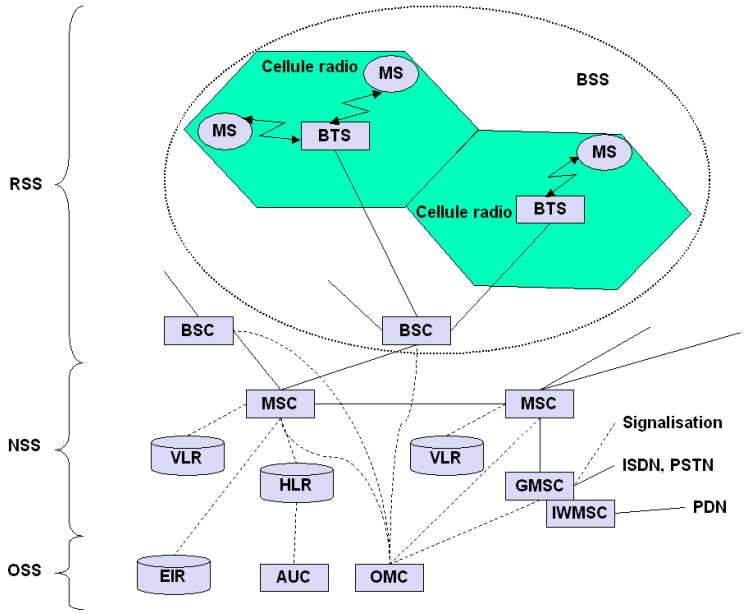


figure 1-1 : infrastructure du réseau GSM.

II.1. Infrastructure du réseau GSM

Dans un réseau GSM peuvent se distinguer trois sous-ensembles : le sous-système radio, le sous-système d'acheminement et le sous-système d'exploitation/maintenance. La figure 1-1 en donne une illustration.

Le sous-système radio appelé RSS (*Radio Sub-System*) ou BSS (*Base Station Sub-system*) est chargé de gérer les ressources radio et d'assurer les transmissions sur ce support. Il est constitué d'un ensemble d'antennes nommées BTS (*Base Transceiver Station*) raccordées à des contrôleurs dénommés BSC (*Base Station Controller*). Chaque BTS est constituée d'un ensemble d'émetteurs-récepteurs radioélectriques échangeant avec les mobiles de la zone géographique (*cellule*) qu'elle couvre. Plusieurs BTS sont rattachées au même BSC et définissent ainsi une *zone de localisation*. Le rôle du BSC est de contrôler cet ensemble de BTS.

Le sous-système d'acheminement, encore connu sous le nom de réseau fixe ou le sigle NSS (*Network Sub-System*), regroupe toutes les fonctions nécessaires à l'établissement des appels et à la gestion de la mobilité. Il est constitué de commutateurs appelés MSC (*Mobile-services Switching Centre ou Mobile Switching Centre*), de routeurs GMSC (*Gateway MSC*) et IWMSC (*InterWorking MSC*) et de bases de données VLR (*Visitor Location Register*) et HLR (*Home Location Register*). Un MSC est chargé de gérer les appels partant du réseau ou y aboutissant. Il est relié à plusieurs BSC. Si généralement un VLR est associé à un MSC, c'est parce qu'il contient des informations de profil et de localisation de tous les mobiles présents dans la zone de couverture d'un MSC. Par contre, le HLR renferme les informations sur le profil et la dernière localisation connue de chacun des abonnés d'un réseau GSM. Le GMSC est employé comme passerelle avec un réseau fixe commuté ou un réseau RNIS (*Réseau Numérique à Intégration de Services*), tandis que le IWMSC sert de passerelle avec les réseaux de données.

Le sous-système d'exploitation/maintenance connu sous les sigles OSS (*Operation Sub-System*) ou OMSS (*Operation and Maintenance Sub-System*) permet à l'opérateur de suivre le fonctionnement du réseau et d'assurer en cas de besoin sa maintenance. On y

retrouve comme dispositifs des centraux tels que l'AUC (*Authentication Centre*) chargé de l'authentification des abonnés d'un réseau, l'OMC (*Operation and Maintenance Centre*) s'occupant du fonctionnement et de la maintenance du réseau, et des bases de données telles que l'EIR (*Equipment Identity Register*) enregistrant les références des terminaux mobiles dont l'accès au réseau doit être refusé.

Face à ce réseau l'usager est muni de son téléphone mobile dans lequel se trouve une carte à puce dite SIM (*Subscriber Identity Mobile*). La SIM est employée principalement comme module d'identification et d'authentification de l'abonné. Elle demeure, dans le monde du GSM, la pièce maîtresse qui va permettre de réaliser des services innovants avec une plus grande valeur ajoutée. C'est elle qui permet à un abonné de consulter son solde téléphonique ou bancaire par interrogation d'une base de données distante via un message court SMS.

Pour plus de détails sur le GSM, nous conseillons l'imposante documentation d'accès libre de la norme GSM. Mais au travers de [LaGoTa00] le lecteur pourra aussi découvrir rapidement les principales caractéristiques de ce système et son fonctionnement.

II.2. Services offerts par le réseau GSM

A l'instar de tout réseau moderne de télécommunications, le réseau GSM offre trois catégories de services : les *services support*, les *télé-services* et les *services supplémentaires*.

Les services support correspondent en gros à la mise à disposition de l'abonné par l'opérateur de "tuyaux" lui permettant d'envoyer entre deux points de l'information utile liée à un service (voix, vidéo, données, etc.), ainsi que toute la signalisation associée. A ces conduits correspondent des paramètres qui vont en définir la QoS (*Quality of Service*) : débit, taux d'erreur, variation du délai, délai d'acheminement de bout en bout, etc. Dans le cas du GSM, ces services support [GSM02.02] se déclinent en trois types :

- le type circuit de données concernant des données transmises entièrement ou partiellement sous forme numérique (débit variant entre 0,3 et 9,6 kbit/s, un type d'accès synchrone ou asynchrone, un mode transparent ou non) ;
- le type accès asynchrone à un réseau de données avec un débit pouvant être asymétrique, compris entre 75 bit/s et 9600 bit/s, et en mode transparent ou non ;
- le type accès synchrone à un réseau de données avec un débit compris entre 2400 bit/s et 9600 bit/s et en mode non transparent uniquement.

Les télé-services se définissent comme les premiers services incluant des fonctions du terminal. Ceux disponibles au niveau du GSM sont : la transmission de la voix, la transmission des messages courts et la télécopie [GSM02.03]. Le service de transmission de la voix se résume à celui des communications téléphoniques déjà offert dans le réseau fixe habituel avec en plus la prise en compte des appels d'urgence même lorsque le téléphone portable n'est pas muni de carte SIM. Le service fax est décomposable en deux catégories : d'une part la transmission alternée voix/fax à 9,6 kbit/s en mode transparent ou non, et d'autre part la transmission automatique fax à 9,6 kbit/s en mode transparent ou non. Quant au service des messages courts, nous y reviendrons en profondeur plus loin étant donné que sa présence permet la réalisation de l'administration OTA.

Les services supplémentaires décrits dans [GSM02.04] [GSM04.10] sont proposés afin d'améliorer les services de base que sont les services support et les télé-services. Ils ne s'emploient donc jamais isolément, mais en conjonction avec un ou plusieurs services support ou télé-services. Ces services supplémentaires introduisent les concepts de permission et d'interdiction d'une opération à un abonné, d'activation ou de désactivation

d'un service pour un abonné, d'interrogation du réseau par l'abonné, de mise en place ou de suppression d'un service dans le réseau par son fournisseur, etc. Certains de ces services ont été standardisés comme l'identification de numéro, le renvoi d'appel, le double appel, la conférence, la facturation et la restriction d'appel. D'autres, qualifiés de *services supplémentaires non structurés*, sont plus spécifiques à l'opérateur, tout en restant conformes à la norme ; ils permettent l'entrée d'ordres à partir du clavier du terminal de l'abonné et à destination du réseau d'une part, et l'obtention en retour d'indications fournies par le réseau au terminal de l'abonné d'autre part.

Ces services sont donc plus ou moins complexes et correspondent à des procédures très structurées. L'approche consistant à composer des procédures élémentaires permet de venir à bout de cette complexité. Ainsi la réalisation du service de réception d'un appel sur un téléphone portable est décomposable en différentes procédures telles que la localisation de la cellule où se trouve le destinataire de l'appel, l'obtention d'un canal sur lequel le réseau peut échanger avec le terminal, l'authentification du terminal initiée par le réseau pour valider l'identité fournie par le terminal, la mise en place du contexte de chiffrement des messages, l'échange des messages, et enfin la libération du canal.

Si de nombreux services sont standardisés, comme par exemple la réception de messages courts (télé-service numéro 21), l'envoi de messages courts (télé-service numéro 22), le téléphone/fax (télé-service numéro 62), le service de diffusion de la parole (télé-service numéro 92), de nombreux services restent encore à mettre en place, soit par les opérateurs de téléphonie, soit par des tiers fournisseurs de services. Les organismes de spécification du GSM ont prévu deux types de transport permettant d'assurer ces services : le transport par SMS et celui par USSD (*Unstructured Supplementary Service Data*). Avant d'aborder la description de ces deux types de transport, décrivons l'environnement de déploiement de la technologie d'administration OTA.

III. Vue d'ensemble du système d'administration OTA

Le système d'administration OTA peut être considéré comme formé de quatre entités communiquant entre elles, en général via les réseaux GSM et Internet : la carte SIM du téléphone mobile, le terminal du téléphone mobile, une passerelle OTA et un serveur OTA. La figure 1-2 en donne une illustration.

La carte SIM représente l'une des terminaisons du système. C'est elle qui doit être modifiée dans le cadre d'une opération d'administration en utilisant l'interface aérienne. Malgré l'apparence, elle fait partie de l'infrastructure de l'opérateur GSM et est simplement mise à la disposition de l'abonné. C'est la raison pour laquelle après sa personnalisation, il est théoriquement impossible d'y faire des modifications si l'on ne dispose pas de privilèges suffisants. Le propriétaire de la SIM – en l'occurrence le MNO ou le MVNO – garde donc un contrôle total de ce qui peut y être téléchargé, comme données et applications, en conformité avec les spécifications de la phase 2+. Nous y reviendrons plus tard.

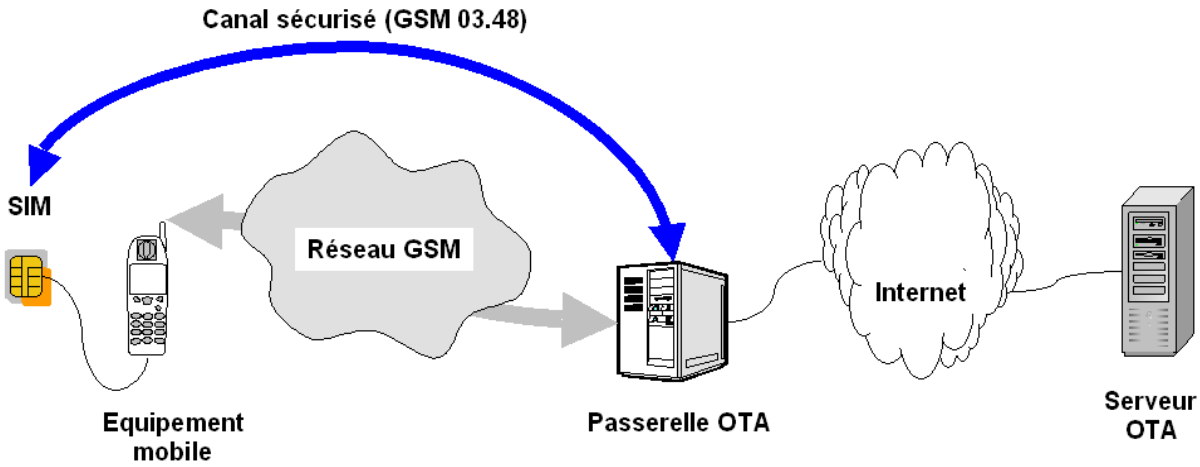


figure 1-2 : vue d'ensemble du système d'administration OTA.

L'équipement mobile accueillant la carte SIM est chargé d'une part de recevoir les messages contenant des données/commandes qu'il transmet à la carte SIM, et d'autre part de recevoir et d'envoyer vers le réseau les messages dont la SIM est l'initiatrice. Du fait des exigences des services OTA, le terminal mobile doit satisfaire aux spécifications de la phase 2+.

La passerelle OTA s'occupe de convertir les demandes de services OTA en un ensemble de commandes (*script*) au format GSM compréhensible par les cartes. Cette traduction doit tenir compte de la diversité des cartes SIM qui existent sur le marché du fait d'une implémentation souvent différente des spécifications GSM. Elle a aussi parfois besoin de récupérer *Over The Air* des informations concernant l'abonné lui-même. Une ou plusieurs transactions correspondant à des échanges de messages à travers un canal sécurisé [GSM03.48] peuvent être nécessaires à la réalisation de l'opération d'administration sur la ou les cartes SIM concernées.

La machine qu'on désigne par "serveur OTA" est la première sollicitée pour administrer des cartes SIM. Elle va formuler, en fonction de la demande reçue, des requêtes permettant d'agir sur une carte SIM particulière, sur un ensemble de cartes SIM d'un type donné ou appartenant à une catégorie de clients, ou encore sur plusieurs catégories de cartes SIM. En étant cliente de la passerelle OTA à laquelle elle fait parvenir ses requêtes, elle peut à son tour faire face aux demandes d'administration OTA qu'elle reçoit en tant que fournisseur de services OTA.

Pour la communication entre ces quatre entités, les réseaux GSM et Internet sont mobilisés. En effet, on a recours au réseau Internet pour supporter les échanges entre le serveur OTA et la passerelle OTA avec l'emploi des services traditionnels Web ou FTP par exemple pour réaliser les échanges. En revanche, entre le téléphone mobile et la passerelle OTA, s'interpose le réseau GSM décrit ci-dessus ; c'est lui qui offre le canal de communication dont l'établissement nécessite l'emploi de l'un des deux types de transport, par SMS ou par USSD, décrits ci-dessous.

IV. Transport par messages courts SMS

IV.1. Généralités

Un message court SMS est une suite d'octets structurée en deux parties [GSM03.40] : un en-tête et une charge utile. L'en-tête qui a au maximum 22 octets, renferme par exemple le type du message (émission ou réception de données d'utilisateurs, de commandes d'utilisateurs, de notifications du réseau, de commandes du réseau), l'adresse de l'expéditeur ou du destinataire du message, l'identificateur du protocole de couche supérieure auquel le message est destiné, le type de codage des données transportées, la date et le délai de livraison du message. La partie correspondant à la charge utile dispose de 140 octets maximum en format non compressé, et 160 maximum dans le cas contraire [GSM03.42] [GSM03.38]. Dans cette dernière partie sont convoyés les messages des utilisateurs, mais aussi les commandes d'administration ou d'accès à des services OTA ; elle peut être structurée afin de recevoir plusieurs messages concaténés.

La présence des services de transport de messages courts est obligatoire au niveau d'un téléphone mobile [GSM02.07]. Ils peuvent être assurés lorsque le téléphone mobile est en veille ou même durant une communication téléphonique, parce qu'ils utilisent essentiellement des canaux de signalisation et non de trafic. Ils mobilisent généralement les équipements suivants au niveau du réseau GSM :

- une entité générique désignée sous le vocable SME (*Short Message Entity*) capable de recevoir ou d'émettre des messages courts. Elle correspond soit à un téléphone mobile, soit au centre d'acheminement de messages courts du réseau GSM, ou encore à un serveur appartenant à un réseau fixe ;
- un centre d'acheminement de messages courts appelé SMS-SC (*SMS Service Centre*) ou simplement SC. Il est capable de recevoir des messages courts et de les envoyer vers leurs destinataires ;
- un routeur capable de retrouver les informations de routage pour l'acheminement des messages courts. Les équipements SMS-GMSC et SMS-IWMSC qualifiés de passerelles jouent ce rôle ;
- un commutateur du genre MSC capable d'être à l'interface du téléphone mobile et du réseau dans le cadre de la réception et de la délivrance des messages courts ;
- des bases de données conservant des informations sur la localisation des destinataires de messages. Le HLR et les VLR sont commis à cette tâche ;
- un téléphone mobile désigné par l'acronyme MS (*Mobile Station*) et qui constitue la cible de l'opération d'administration.

Les différents équipements impliqués dans le transport des messages courts sont représentés à la figure 1-3.

Sur le segment 1 transitent des messages courts de deux sortes : ceux envoyés du SC vers la passerelle GMSC et à destination d'un mobile d'une part, et ceux envoyés de la passerelle IWMSC vers le SC d'autre part. Sur ce segment circulent également des rapports de livraison ou d'erreur entre le SC et les passerelles. Le segment 2 illustre la recherche d'informations de routage (adresse du MSC ou du SC auquel transmettre le message court) auprès du HLR. Le tronçon 3 correspond principalement au transfert de messages courts de la passerelle GMSC au MSC dans le cas d'un service de transport à destination d'un mobile, et du MSC vers le IWMSC dans le cas d'un service ayant pour origine un mobile. Il va sans dire que ce segment supporte aussi le transfert des rapports de livraison et d'échec. Grâce aux échanges sur le segment 4, le MSC peut se renseigner sur la localisation précise du mobile destinataire du message ou vérifier que ce service ne souffre

d'aucune restriction de la part de l'opérateur. Finalement le segment 5 représente la livraison ou l'expédition du message court et le retour d'un rapport de livraison ou d'erreur.

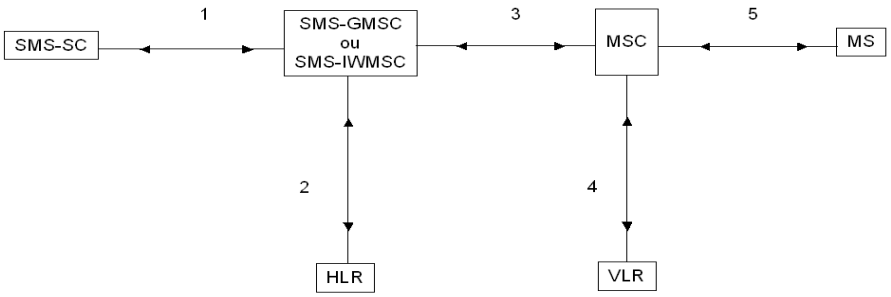


figure 1-3 : contexte général d'échange de messages dans le réseau GSM.

Dans le service de transport par messages courts, deux catégories se distinguent et concernent directement l'abonné : le type *service de transport par messages courts en point à point*, et le type *service de transport par messages courts en diffusion vers des mobiles*. Leur fonctionnement est décrit dans [GSM02.03].

Le transport par messages courts en diffusion vers des mobiles, connu sous l'appellation CBS (*Cell Broadcast Service*), est décrit dans [GSM03.41]. Il permet de transmettre un message court de 93 caractères maximum d'un centre de messages SMS vers un ensemble de téléphones mobiles présents dans la couverture géographique d'une BTS. Il est généralement employé par un diffuseur d'informations en direction de clients GSM d'une localité donnée, et très rarement pour une administration OTA.

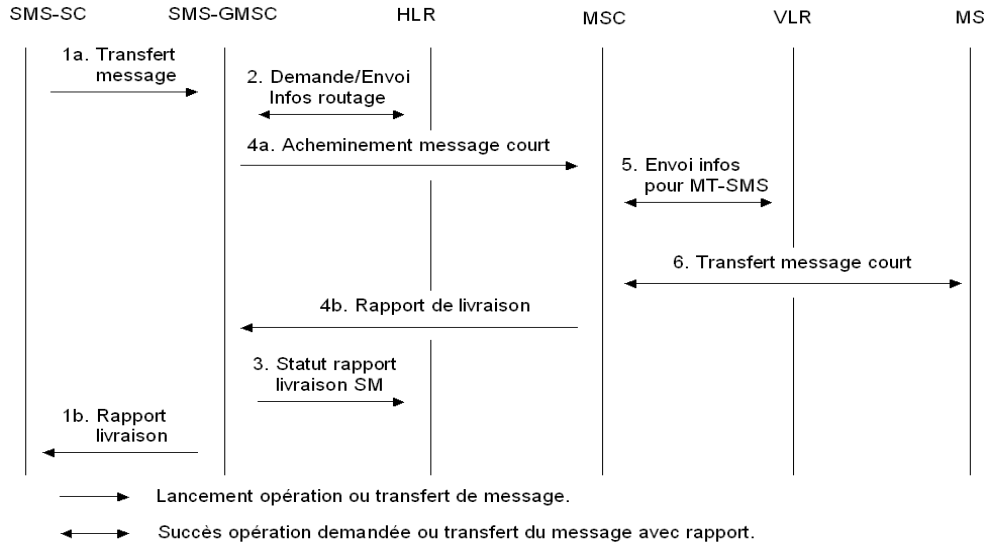


figure 1-4 : tentative réussie de livraison d'un message court à un mobile.

Le type de transport par messages courts point à point représente le moyen largement employé lors de l'administration OTA. Selon l'origine de la demande de service de transport par messages courts, deux catégories peuvent être distinguées : la catégorie des messages courts arrivant à un mobile en point à point, et celle des messages courts provenant d'un mobile en point à point.

La catégorie des services de transport par messages courts vers un mobile en point à point, en abrégé SM MT (*Short Message Mobile Terminated Point to Point*), est d'abord employée pour transférer tout message court ou compte-rendu du SC vers le MS. Ces services peuvent aussi être employés pour retourner un rapport contenant le résultat d'une tentative d'envoi d'un message au SC. La catégorie des services de transport par messages courts provenant d'un mobile en point à point dénommé SM MO (*Short Message Mobile Originated Point to Point*) sert plutôt à l'acheminement des messages courts d'un MS vers un SC. Elle peut aussi servir à renvoyer au MS un rapport contenant le résultat d'une tentative de transfert d'un message court.

La figure 1-4 présente un essai réussi de transfert d'un message court vers un mobile. Il y apparaît un transfert direct du message court au mobile par le MSC, dès que celui-ci en connaît la localisation précise. Un rapport de cette remise est fait à la passerelle SMS-GMSC qui le transmet au centre d'acheminement initiateur du transfert. En revanche, la survenue d'une erreur dans le parcours allant du centre d'acheminement au mobile occasionne l'échec du transfert ; ceci est illustré à la figure 1-5 où une erreur se produit au moment de la délivrance du message sur l'interface radio.

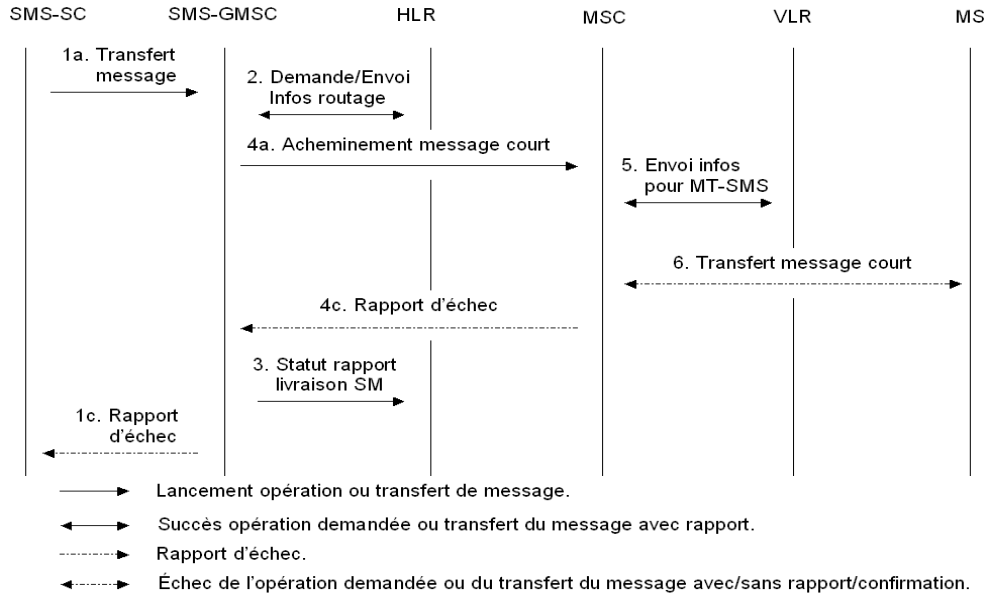


figure 1-5 : tentative infructueuse de livraison d'un message court à un mobile.

IV.2. Architecture protocolaire sous-jacente

Selon les spécifications [GSM03.40], la réalisation d'une application s'appuyant sur des services de transport par messages courts suppose la présence d'une pile protocolaire (voir figure 1-6) dans les équipements d'extrémité concernés par le service et contenant : une couche liaison dénommée SM-LL *(Short Message Lower Layers*) cachant les aspects basiques des équipements intervenant dans la réalisation du service, une couche de relais dénommée SM-RL (*Short Message Relay Layer*) mettant en œuvre un protocole SM-RP (*Short Message Relay Protocol*), une couche Transport appelée SM-TL (*Short Message Transfer Layer*) associée à un protocole du nom de SM-TP (*Short Message Transfer Protocol*) et une couche Application appelée SM-AL (*Short Message Application Layer*). Nous nous limitons à la présentation de la couche Transport. En effet, les entités de la couche Application ne sont pas spécifiées dans le GSM, et, toutes les autres couches correspondent à celles

employées dans le cadre de la réalisation du service d'appel téléphonique classique [LaGoTa00].

La couche Transport SM-TL est chargée de rendre six types de services à la couche supérieure Application, des services destinés à l'acheminement de données de bout en bout. Les TPDU (*Transport Packet Data Unit*) qui les réalisent se nomment SMS-DELIVER, SMS-STATUS-REPORT, SMS-DELIVER-REPORT, SMS-SUBMIT, SMS-COMMAND et SMS-STATUS-REPORT. A la
figure 1-7 sont représentées les entités impliquées dans la fourniture de ces différents services de transport.

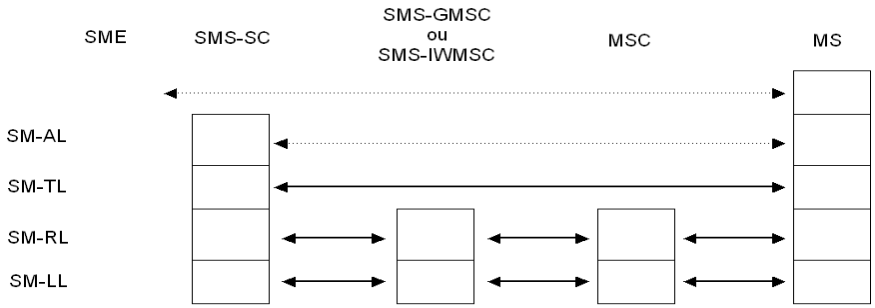


figure 1-6 : pile protocolaire utilisée lors d'un transport par messages courts SMS.

Pour l'acheminement d'un message d'un SC à destination d'un MS, le TPDU de type SMS-DELIVER est employé. Il contient des informations décrivant le type de message, l'adresse du SME à l'origine du message, des indications concernant la date et l'heure de réception du message par le SC, une indication de l'existence éventuelle d'une "route" pour l'acheminement de la réponse au message, etc.

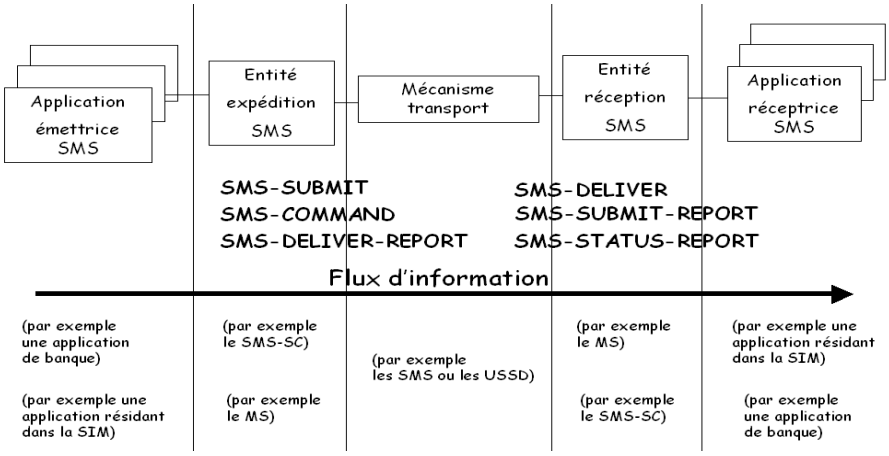


figure 1-7 : entités impliquées dans le transport par SMS.

La transmission d'une cause d'échec éventuelle ou d'un accusé de réception positif ou négatif faisant suite à un paquet SMS-DELIVER ou SMS-STATUS-REPORT est assurée grâce à un TPDU du type SMS-DELIVER-REPORT envoyé du MS vers le SC.

Pour l'acheminement d'un message du MS à destination du SC, le TPDU SMS-SUBMIT est employé. Il contient l'adresse du SME destinataire, un identificateur du

paquet, et certaines des informations contenues dans le paquet de type SMS-DELIVER déjà présenté.

Pour le transport du SC vers le MS, de l'indication d'une cause d'échec éventuelle ou d'un accusé de réception positif ou négatif reçu après l'envoi d'un paquet de type SMS-SUBMIT ou SMS-COMMAND, le TPDU utilisé est du type SMS-SUBMIT-REPORT.

Le TPDU SMS-COMMAND est utilisé pour acheminer une commande SMS en provenance du MS et à destination du SC afin d'agir sur un message court, désigné, présent au niveau du SC (destruction, modification du bit de demande de rapport, etc.). Il contient, entre autres champs, l'identificateur du message court sur lequel la commande doit porter, l'identificateur de la commande elle-même, un indicateur de demande éventuelle de rapport à la suite de l'exécution de la commande.

Enfin, lorsqu'un rapport d'exécution est à renvoyer et concerne la dernière commande SMS-COMMAND ou SMS-SUBMIT reçue en provenance du MS, c'est le TPDU de type SMS-STATUS-REPORT envoyé du SC vers le MS qui est employé. Il renferme par exemple des indications permettant d'identifier la dernière commande concernée par le rapport, l'indication de la date et de l'heure où la commande à rapporter a été reçue par le SC.

Ces services sont offerts à une application de la couche protocolaire supérieure pour l'acheminement de données pouvant correspondre au texte d'un message, ou encore à des commandes/réponses [GSM11.14] [GSM03.19]. Ce type de transport va permettre à deux applications distantes de communiquer, dans le but de fournir un service particulier à l'utilisateur ; la consultation de son solde bancaire ou l'activation de nouveaux services accessibles à l'abonné en sont deux illustrations.

V. Transport par USSD

Les services supplémentaires employant des données non structurées sont définis comme des services supplémentaires (services par définition modifiant des services de base) dont le fonctionnement est caractérisé par des échanges de chaînes de caractères entrées par l'usager et directement envoyées de façon transparente au réseau, et dans le sens inverse de messages envoyés par le réseau et directement affichés à l'usager. Nous estimons que ce type de services de transport peut venir en complément des services de transport par messages courts, surtout dans les situations où une interactivité avec l'usager est requise.

A la différence du type de transport par messages courts, la présence du type de transport via des données d'un service supplémentaire non structurées USSD (*Unstructured Supplementary Service Data*) est optionnelle au niveau d'un téléphone portable selon [GSM02.07]. Ce type de transport est décrit dans [GSM02.90] et [GSM03.90]. Il permet, à l'usager d'un téléphone portable et à une application donnée d'un réseau d'opérateur, de communiquer de façon transparente aussi bien vis-à-vis du téléphone portable que des équipements réseaux intermédiaires. Le schéma de la figure 1-8 présente l'environnement dans lequel ce type de transport se déroule ; un gestionnaire de données USSD est présent aussi bien dans le réseau que sur le mobile.

Un transport par USSD peut être initié soit à partir du téléphone portable de l'utilisateur, soit par le réseau GSM au sein duquel l'application est hébergée. Dans le réseau l'application est installée soit sur le MSC, soit sur le HLR ou encore sur le VLR. C'est aux fournisseurs de services et aux opérateurs qu'incombe la tâche de définir la localisation, la nature et le contenu des applications traitant les USSD.

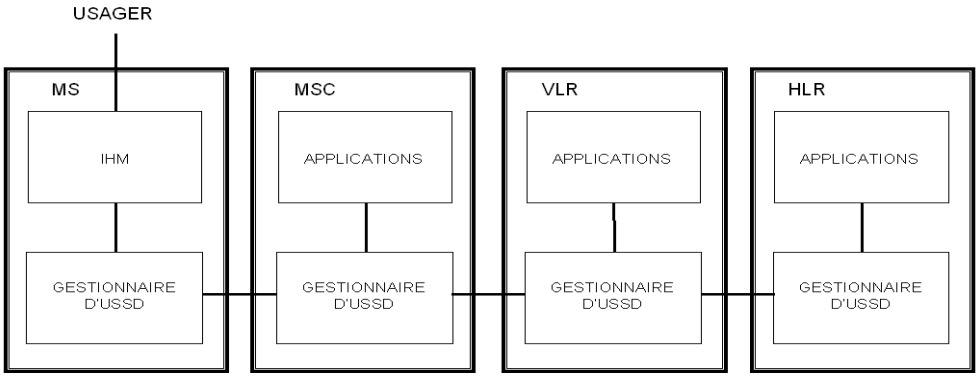


figure 1-8 : contexte de traitement d'une donnée USSD.

Quand le réseau est à l'origine d'un service de transport par USSD, ce qu'il envoie correspond soit à une requête (demande de fourniture de certaines informations faite au téléphone portable) soit à une notification (compte-rendu fait au téléphone portable) qui peuvent survenir à n'importe quel moment. Démarré par le HLR, ce service transite d'abord par le VLR, puis par le MSC avant d'atteindre le téléphone portable. Initiée par le VLR, l'opération qui utilise le transport USSD ne nécessite que le transit par le MSC avant d'aboutir au téléphone portable. Finalement, si c'est au niveau du MSC que l'opération basée sur les USSD est déclenchée, ses données sont envoyées au téléphone portable sans intermédiaire. Dans chacun des cas cités, l'obtention de la réponse déclenche la libération des ressources par le même chemin que celui de leur mobilisation.

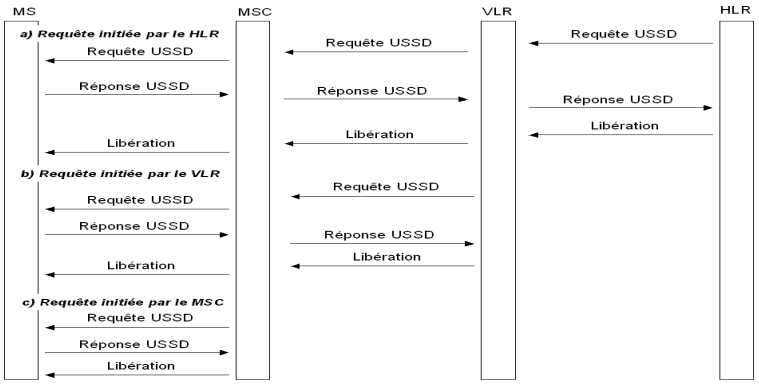


figure 1-9 : cas de requête USSD émanant du réseau et portant sur une simple opération.

A la figure 1-9 sont présentés pour la réalisation d'une simple opération, trois cas de requêtes utilisant le transport par USSD. Elles sont démarrées respectivement par le HLR, le VLR et le MSC. Un scénario semblable aurait été exécuté s'il s'était agi d'une notification en provenance du réseau.

Dans le cas d'un service initié par le téléphone portable, celui-ci est d'abord démarré par l'usager agissant sur son téléphone portable qui à son tour transfère la requête au MSC.

Si le MSC constate que la requête contient un code de service lié au réseau où a souscrit l'abonné, il démarre une transaction avec le VLR auquel est transférée la requête. Pour tous les échanges en rapport avec cette opération USSD, ce MSC jouera un rôle de

relais transparent. Au cas où un échec survient au moment du transfert de la requête au VLR, une indication d'erreur est retournée par le MSC au téléphone portable. En revanche si la requête reçue par le MSC ne contient pas de code de service de réseau, elle est traitée par l'application correspondante au niveau du MSC (figure 1-10, cas a).

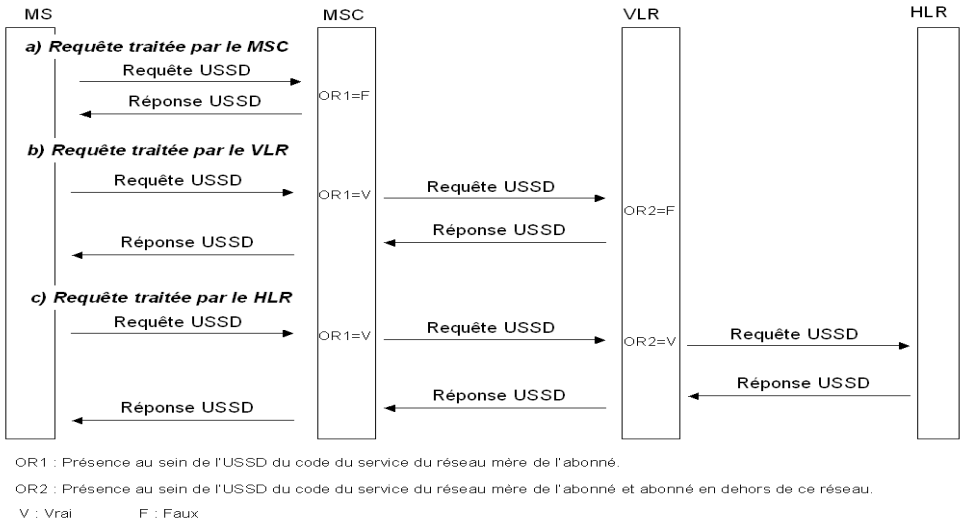


figure 1-10 : cas simples d'initiation par le mobile d'un service de transport par USSD.

Lorsque le VLR reçoit une requête contenant un code de service du réseau auquel l'abonné a souscrit et que l'abonné est dans une zone géographique couverte par son opérateur, la requête est traitée localement par l'application correspondante (cas b de la figure 1-10). C'est aussi un traitement local de la requête qui est effectué lorsqu'elle est reçue sans code de service. Si la requête reçue par le VLR contient soit un code de service du réseau auquel l'abonné a souscrit et que l'abonné est en dehors de la zone géographique couverte par son opérateur, soit un alphabet étranger au VLR, alors elle est acheminée au niveau du HLR qui traite toujours en local toutes les requêtes qui lui parviennent, comme l'indique le cas c de la figure 1-10.

Il arrive parfois que la requête adressée à l'application employant une donnée USSD dans le réseau soit à son tour à l'origine d'une nouvelle requête adressée cette fois au mobile (complément d'information à fournir par l'usager par exemple). De nombreuses transactions intermédiaires impliquant des USSD peuvent alors avoir lieu avant que la réponse à la requête initiale ne soit délivrée par l'application au mobile.

VI. Conclusion

Dans ce chapitre nous avons voulu d'abord présenter les entités fonctionnelles intervenant dans la réalisation d'une administration OTA. Ensuite, l'essentiel de l'effort de description a porté sur les deux principaux services de transport offerts par le GSM pour l'échange d'informations entre le réseau et le téléphone portable de l'abonné : le service de transport par messages courts SMS, et le service de transport par USSD.

Ce chapitre a permis d'appréhender ce qui se passe à la frontière du téléphone portable dans le cadre d'une administration OTA. La suite du travail concernera les échanges ayant lieu à l'intérieur du téléphone portable lui-même.

Chapitre 2 : Mécanismes internes au téléphone portable dans l'administration OTA

Nous avons au chapitre précédent donné une présentation des interactions entre le réseau et le téléphone mobile, des interactions qui emploient dans le cadre de l'administration OTA, les services de transport par messages courts et par données de services supplémentaires non structurées. L'objectif ici est d'aller un peu plus loin dans la compréhension de ce qui a lieu au sein du téléphone mobile lui-même. Pour cela, nous débutons le chapitre avec une présentation des entités fonctionnelles d'un téléphone mobile. Puis, nous décrivons les mécanismes employés pour assurer d'une part, une cohérence dans la réalisation des services de base et des services supplémentaires standardisés, d'autre part une ouverture vis-à-vis des futurs services supplémentaires qui, avec le temps, se mettront en place pour répondre aux besoins des utilisateurs.

I. Les entités fonctionnelles du téléphone portable

Comme le montre la
figure 2-1 deux composants se distinguent dans un téléphone portable désigné MS (*Mobile Station*) dans la documentation du GSM : une carte SIM (*Subscriber Identity Module*) et un équipement mobile ME (*Mobile Equipement*).

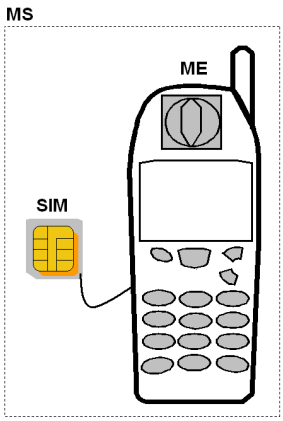


figure 2-1 : composants d'un téléphone portable GSM.

La carte SIM représente le cœur du téléphone portable. Elle renferme les informations identifiant et authentifiant l'abonné dans le réseau GSM, et de ce fait constitue la cible des opérations courantes d'administration OTA. Nous y reviendrons plus loin.

L'équipement mobile correspond au terminal accueillant la carte SIM. Il demeure l'entité du téléphone portable en contact physique avec l'utilisateur. Par son intermédiaire se réalisent toutes les interactions entre l'usager et le reste du système. On y retrouve des dispositifs IHM (Interface Homme-Machine), des entités de traitement des signaux radio, des composants de traitement de signaux numériques, l'alimentation électrique, etc.

Les deux composants que sont la carte SIM et l'équipement mobile fonctionnent en tandem. Chacun dispose de ses propres capacités de traitement et de stockage plus ou moins importantes. Sur l'équipement mobile est installé un système d'exploitation, comme

c'est le cas au niveau de la carte SIM. L'accès à certains nouveaux services est conditionné par la présence de certains environnements des deux côtés comme dans un modèle client/serveur.

Afin d'assurer un fonctionnement harmonieux de cet ensemble, un protocole d'échange est défini entre ces deux composants : c'est le protocole T=0 défini par la norme ISO/IEC 7816-3 [7816-3_97] qui gère les interactions SIM-ME ; il est étudié à la section suivante.

II. Le protocole d'échanges SIM – ME

Le protocole T=0 définit une transmission asynchrone par caractère entre la SIM et l'équipement mobile. Ces échanges s'effectuent selon un modèle transactionnel requête/réponse dans lequel l'équipement mobile est seul habilité à envoyer des requêtes, et la SIM toujours en devoir d'y répondre. Autrement dit, c'est toujours à l'initiative de l'équipement mobile qu'une opération se déroule au niveau du téléphone portable. Les commandes/réponses sont appelées APDU (*Application Protocol Data Unit*) [7816-4_95] et se présentent selon des formats illustrés à la figure 2-2.

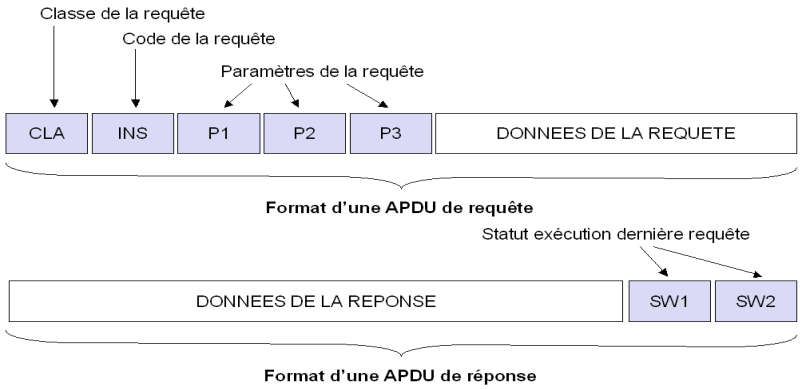


figure 2-2 : formats types d'APDU échangés entre la SIM et l'équipement mobile.

Dans une commande, le champ CLA (*Class*) tient sur un octet et désigne la classe de la commande ; par exemple la valeur hexadécimale "A0" désigne la catégorie des commandes du GSM. L'octet INS (*Instruction*) est associé au code de la commande, et les trois octets P1, P2 et P3 (*Parameter*) correspondent aux paramètres de la commande, avec en particulier P3 associé au nombre d'octets de données figurant dans la commande.

Dans une réponse les octets SW1 et SW2 sont les indicateurs d'état décrivant la manière dont la dernière commande au niveau de la SIM a été réalisée. Le tableau 2-1 présente des exemples de transactions réussies entre la SIM et l'équipement mobile. La taille des données d'une réponse correspond normalement à une longueur fournie dans la requête à laquelle elle est associée. Par la transaction 3, l'exécution d'une commande ne transportant aucune donnée en son sein est demandée à la puce. La puce la réalise correctement et avertit l'équipement mobile de la disponibilité de *lgr1* octets correspondant au résultat à récupérer. En démarrant la transaction 3', l'équipement mobile par la commande GET RESPONSE *lgr2* demande à la carte SIM de lui renvoyer les *lgr2* (*lgr2*≤*lgr1*) octets correspondant au résultat de la précédente commande. La carte SIM répond à cette dernière commande par l'envoi des *lgr2* octets auxquels est jointe la paire d'octets de code hexadécimal '90' '00' signifiant le bon déroulement de la commande.

La présence de la SIM est indispensable pour tous les services fondés sur l'appel, sauf dans le cas des appels en urgence. Pour en vérifier la présence, l'équipement mobile

interroge périodiquement, environ toutes les trente secondes, la SIM en utilisant la commande STATUS. L'absence de réponse de la part de la SIM est synonyme d'absence de carte SIM dans le téléphone mobile.

tableau 2-1 : exemples d'interactions SIM – ME basiques réussies.

N° transaction	Commande			Réponse		Commentaires sur la transaction
	Valeur de P3	Données		Données	Statut SW1 SW2	
1	'00'	Aucune	→ ←	Aucune	'90' '00'	Sans donnée contenue ni dans la commande ni la réponse
2	*lgr1*	Aucune	→ ←	*lgr1* octets	'90' '00'	Sans donnée contenue dans la commande mais avec *lgr1* octets dans la réponse
3	'00'	Aucune	→ ←	Aucune	'9F' *lgr1*	Sans donnée en entrée mais *lgr1* octets sont à récupérer par GET RESPONSE
3'	GET RESPONSE avec *lgr2 (≤lgr1)*	Aucune	→ ←	*lgr2* octets	'90' '00'	Récupération correcte partielle ou totale des données
4	*lgr*	*lgr* octets	→ ←	Aucune	'90' '00'	*lgr* octets contenus dans la commande et aucune donnée dans la réponse
5	*lgr4*	*lgr4* octets	→ ←	Aucune	'9F' *lgr5*	*lgr4* octets contenus dans la commande avec *lgr5* octets à récupérer par GET RESPONSE
5'	GET RESPONSE avec *lgr6 (≤lgr5)*	Aucune	→ ←	*lgr6* octets	'90' '00'	Récupération correcte partielle ou totale des données

En vue d'informer l'équipement mobile des services que peut assurer la SIM, une table de services est présente dans la SIM. Elle indique d'une part les services réalisables par la SIM, et d'autre part ceux mis à la disposition de l'abonné dans le cadre du contrat souscrit avec l'opérateur. Bien évidemment ne peut être utilisé par un abonné qu'un service réalisable par la SIM, et tous les services réalisables par la SIM ne sont pas systématiquement rendus disponibles à tout client GSM. De même dans la SIM se trouve un fichier indiquant la phase GSM correspondant à la SIM ; ses informations complètent celles de la table des services sur les capacités de la SIM.

Les requêtes que l'équipement mobile peut adresser à la SIM peuvent être regroupées en sept catégories qui concernent :

- la réalisation des entrées-sorties sur les fichiers de la SIM (sélection, récupération du statut, lecture, écriture, positionnement dans le fichier) ;
- l'administration de la SIM elle-même (initialisation, contrôle de la présence de la SIM, scrutation dans le cadre de la SIM proactive, langage utilisé pour l'interface homme-machine, etc.) ;
- la gestion des droits et autorisations sur les objets de la SIM (vérification, modification, activation, désactivation, blocage, déblocage du code PIN[5] du porteur de la carte, validation et invalidation d'un fichier) ;
- la sécurité de l'accès au réseau GSM (calcul de clés cryptographiques, lecture de l'IMSI, réseaux interdits, etc.) ;
- l'abonnement de l'usager (numéros de téléphone, facturation, choix de réseau, nom du fournisseur de services, etc.) ;
- la commande par la SIM des opérations à l'équipement mobile au travers de l'environnement optionnel appelé SAT (*SIM Application Toolkit*) ;

[5] *Personal Identity Number* est l'ancien nom du CHV (*Card Holder Verification information*) correspondant au code secret défini par l'abonné, et stocké sur la carte SIM pour l'accès aux services.

- la mise en place de futurs services par des fournisseurs de services appelés MExE (*Mobile Exécution Environment*).

Les cinq premières catégories de requêtes sont employées dans les services classiques d'appel téléphonique et d'envoi/réception de messages courts par l'usager. Les deux dernières catégories sont particulièrement intéressantes puisque prévues dès le départ par les normalisateurs du GSM pour favoriser le développement de nouveaux services supplémentaires accessibles via le téléphone portable ; nous y reviendrons plus loin.

III. SIM proactive et SIM Application Toolkit (SAT)

La SIM proactive est la SIM capable de demander à l'équipement mobile l'exécution d'une commande particulière dont les résultats devront lui être retournés. Cette qualité de la SIM ne modifie pas le mode initial de fonctionnement du téléphone mobile, en ce sens que la commande dont l'exécution est demandée à l'équipement mobile lui est transmise comme réponse à l'exécution d'une commande FETCH faisant partie du cadre optionnel *SIM Application Toolkit* (SAT) [GSM11.14]. Pour illustrer ce fonctionnement, considérons l'exemple suivant.

Supposons que la SIM ait besoin de faire réaliser par l'équipement mobile une opération spécifique, par exemple l'affichage d'un message. Elle fait connaître son "intention" lors de la délivrance d'une réponse contenant '91' 'XX' comme valeurs de SW1 SW2, ce qui en termes de discours humain peut être traduit par "*dernière commande bien exécutée mais permets-moi de soumettre une commande de XX octets à mon tour*". A la réception de cette valeur de statut, l'équipement mobile envoie dès que possible une commande spéciale dénommée FETCH employant comme paramètre en entrée XX. La carte SIM y répond avec, dans le champ réservé aux données, la fourniture de la commande *lambda* de XX octets dont elle demande l'exécution. Dès réception de la réponse au FETCH, l'équipement mobile exécute la commande *lambda* et renvoie son résultat à la SIM au travers d'une commande TERMINAL RESPONSE. A cette dernière commande la SIM répond par le statut '90' '00' signifiant qu'elle est satisfaite du travail de l'équipement mobile. Vu l'importance qu'ont les valeurs codant le statut accompagnant une réponse délivrée par la SIM, ont été consignées dans le tableau 2-2 les principales valeurs de SW1 et SW2.

La présence de la SAT dans le téléphone portable étend la liste des commandes régissant les interactions entre la SIM et l'équipement mobile comme le montre le tableau 2-3.

tableau 2-2 : principales valeurs de retour associées à l'exécution d'une commande.

SW1	SW2	Description du statut
Cas de réponses obtenues après une exécution correcte de la commande		
'90'	'00'	Fin correcte de la commande et aucune donnée en résultat à renvoyer
'91'	'XX'	Fin correcte de la commande avec le désir de la part de la SIM de lancer une commande proactive dont la taille en octets est en hexadécimal 'XX'
'9F'	'XX'	Fin correcte de la commande avec des données en résultat de taille 'XX' en hexadécimal disponibles et récupérables via un commande GET RESPONSE
'9E'	'XX'	En réponse à une commande TERMINAL PROFILE, la SIM indique qu'elle sait gérer l'indicateur d'état codé '9EXX' en hexadécimal. En réponse à une commande ENVELOPE, la SIM indique en retour la disponibilité de données de taille 'XX' en hexadécimal
Cas de réponses obtenues après une erreur d'exécution de la commande		
'93'	'00'	La SAT est occupée. La commande ne peut être exécutée actuellement, mais pourrait l'être ultérieurement
'92'	'YY'	Problèmes de gestion de la mémoire
'94'	'YY'	Mauvais adressage lors de la manipulation du fichier
'98'	'YY'	Problèmes de droits d'accès
'6X'	'YY'	Erreur en rapport avec la codification de la commande

tableau 2-3 : nouvelles commandes SAT à la disposition de l'équipement mobile.

Commande	Description
FETCH	Demande de transfert d'une commande SAT de la SIM vers l'équipement mobile en vue de son exécution
TERMINAL RESPONSE	Transfert vers la SIM de la réponse associée à l'exécution de la précédente commande SAT précédemment FETCHée
TERMINAL PROFILE	Transfert des capacités fonctionnelles en matière de SAT de l'équipement mobile vers la SIM
ENVELOPE	Transfert de données vers la SAT dans la SIM

tableau 2-4 : exemples de transactions entre équipement mobile et SIM proactive.

N° transaction	Commande			Réponse		Commentaires sur la transaction
	Valeur de P3	Données		Données	Statut SW1/SW2	
1a	*lg1*	Aucune	→ ←	*lg1* octets	'91' *lg2*	Sans donnée en entrée, *lg1* octets en résultat et désir d'exécution d'une commande SIM proactive de *lg2* octets
1b	FETCH avec comme paramètre *lg2*	Aucune	→ ←	*lg2* octets liés à la commande proactive	'90' '00'	Sans donnée en entrée mais *lg2* octets en résultat codant la commande SIM proactive
1c	TERMINAL RESPONSE avec comme paramètre *lg3*	*lg3* octets	→ ←	Aucune	'90' '00'	*lg3* octets en entrée associés au résultat de l'exécution correcte de la dernière commande FETCHée
2a	*lg4*	*lgr4* octets	→ ←	Aucune	'91' *lg5*	*lg4* octets en entrée, sans donnée en résultat et désir d'exécution d'une commande SIM proactive de *lg5* octets
2b	FETCH avec comme paramètre *lg5*	Aucune	→ ←	*lg5* octets liés à la commande proactive	'90' '00'	Sans donnée en entrée mais *lg5* octets en résultat codant la commande SIM proactive
2c	TERMINAL RESPONSE avec comme paramètre *lg6*	*lg6* octets	→ ←	Aucune	'91' *lg5*	Problème survenue lors de l'exécution de la commande proactive et demande réitérée
2d	FETCH avec comme paramètre *lg5*	Aucune	→ ←	*lg5* octets liés à la commande proactive	'90' '00'	Sans donnée en entrée mais *lg5* octets en résultat codant la commande SIM proactive
2e	TERMINAL RESPONSE avec comme paramètre *lg7*	*lg7* octets	→ ←	Aucune	'90' '00'	*lg7* octets en entrée associés au résultat de l'exécution correcte de la dernière commande FETCHée

Le tableau 2-4 illustre les échanges SIM – ME dans un environnement de carte SIM proactive par huit transactions exploitant les valeurs de l'octet P3 (commande) et du statut SW1/SW2 (réponse).

L'équipement mobile peut prendre connaissance des capacités de la SIM en consultant dans la SIM les fichiers décrivant la phase et les services qui lui correspondent. De même la SIM peut connaître les capacités de l'équipement mobile qui l'héberge, grâce à la commande TERMINAL PROFILE qui s'exécute durant toute phase d'initialisation de la SIM. Ces capacités figurant dans le profil transféré à la SIM sont décrites par les commandes dans le tableau 2-5.

En plus de la capacité offerte à la SIM de demander l'exécution d'une commande à l'équipement mobile, on retrouve dans l'environnement de la SAT la possibilité de télécharger dans la SIM des données provenant d'ailleurs ; c'est la commande ENVELOPE. Elle permet, si le service est disponible sur la SIM et utilisable par l'abonné, d'envoyer vers la SIM les informations concernant un message court en rédaction, grâce à sa forme ENVELOPE (*MO SHORT MESSAGE CONTROL*). Deux autres formes, ENVELOPE (*SMS-PP DOWNLOAD*) et ENVELOPE (*CELL BROADCAST DOWNLOAD*), permettent de transmettre à la SIM un message reçu du réseau et contenant une requête ou une donnée d'administration OTA.

tableau 2-5 : commandes dont l'exécution peut être demandée par la SIM proactive.

Commande	Description
REFRESH	Réinitialisation de la SIM
MORE TIME	Demande de temps supplémentaire pour le traitement de la commande en cours
POLL INTERVAL	Négociation de la périodicité des commandes STATUS quand le mobile est en veille
POLLING OFF	Arrêt de la scrutation de la SIM
SET EVENT LIST	Fourniture de la liste d'événements déclencheurs
SET UP CALL	Démarrage d'un appel
SEND SHORT MESSAGE	Envoi d'un message court
SEND USSD	Envoi de données non structurées dans le cadre d'un service supplémentaire
SEND SS	Demande de lancement d'un service supplémentaire
PLAY TONE	Jeu d'une sonorité/mélodie
DISPLAY TEXT	Affichage d'un message
GET INKEY	Affichage d'un message ou d'une icône et récupération d'un caractère du clavier
GET INPUT	Affichage d'un message ou d'une icône et récupération d'une chaîne de caractères au clavier
PROVIDE LOCAL INFORMATION	Fourniture d'informations sur la localisation du mobile
SET UP MENU	Fourniture d'une liste d'éléments d'un menu
SELECT ITEM	Présentation d'une liste d'options offertes à l'usager et récupération de son choix
LANGUAGE NOTIFICATION	Indication du langage courant employé au niveau de l'interface homme-machine
SETUP IDLE MODE TEXT	Mise en place d'un texte affiché en mode veille
TIMER MANAGEMENT	Demande de gestion d'un temporisateur
SEND DTMF	Demande d'envoi d'une sonnerie après établissement d'une connexion

La SIM peut confier à l'équipement mobile une liste d'événements à surveiller au moyen de la commande proactive SET EVENT LIST. Lorsque l'un des événements de cette liste survient au niveau de l'équipement mobile, celui-ci en informe la SIM dès que possible en envoyant une commande ENVELOPE adéquate ; l'événement est conservé dans une file d'attente FIFO lorsque l'interface SIM-ME est déjà occupée. Et les principales sortes de commandes ENVELOPE qui peuvent être envoyées à la SIM sont :

- ENVELOPE (*EVENT DOWNLOAD - MT call*) à l'arrivée d'un appel ;
- ENVELOPE (*EVENT DOWNLOAD – call connected*) en cas de réalisation d'une connexion ;
- ENVELOPE (*EVENT DOWNLOAD – call disconnected*) ;
- ENVELOPE (*EVENT DOWNLOAD – Location status*) en cas de fin de connexion ;
- ENVELOPE (*EVENT DOWNLOAD – User activity*) au cas où l'usager appuie sur une touche ;

- ENVELOPE (*EVENT DOWNLOAD – Idle screen*) au cas où le mobile passe en mode veille ;
- ENVELOPE (*EVENT DOWNLOAD – Language selection*) au cas où l'usager choisit un langage de communication avec le mobile.

La commande proactive SEND SHORT MESSAGE permet à la SIM de commander l'envoi vers le réseau d'un message court pouvant contenir soit des données de l'usager sous forme compressée ou normale, soit une réponse/requête destinée au serveur d'administration OTA. Cet envoi de message court par la SIM peut être complètement transparent à l'utilisateur, avec un compte-rendu de l'envoi du message par l'équipement mobile fait à la SIM au travers d'une commande TERMINAL RESPONSE. La commande proactive SEND USSD fonctionne à la manière de la commande SEND SHORT MESSAGE mais avec des données de services supplémentaires non structurées. Il s'agit d'un dialogue en direct, entre l'usager et une application abritée au sein d'un équipement du réseau GSM.

La commande proactive SEND SS permet de demander la réalisation d'un service supplémentaire comme le transfert d'appel, la restriction d'appel, l'activation du contrôle d'un service par mot de passe, la présentation du numéro d'appel, l'enregistrement ou la suppression d'un service du réseau, le démarrage d'un service supplémentaire donné dans le réseau, etc. [GSM02.04] [GSM04.80].

L'exemple d'un opérateur ou d'un fournisseur de services désirant modifier le contexte de réalisation d'un service donné chez un abonné illustre la manière dont ces capacités inhérentes à la présence de la SAT pourraient être employées. En effet, du réseau arrive un message SMS d'administration en direction du téléphone portable de l'abonné (service transport SMS MT point à point). Ce message SMS est reçu par l'équipement mobile qui le transmet à la carte SIM par une commande ENVELOPE. La carte SIM via la SAT exécute alors l'application correspondant au message reçu en fonction du profil de l'abonné. A la fin du traitement, le résultat est transmis à l'équipement mobile. Celui-ci envoie alors à destination du réseau un SMS d'acquittement dans lequel figure le résultat obtenu par la SIM.

IV. Les environnements d'exécution mobile

Les environnements d'exécution mobile ont été prévus afin d'étendre encore le champ des services de la SIM. Comme l'environnement SAT, l'environnement d'exécution mobile MExE (*Mobile Execution Environment*) est optionnel pour le téléphone portable. Il doit permettre de fournir à l'utilisateur des services variés, quel que soit le réseau d'accès qu'il emploie et quel que soit le terminal à partir duquel il veut y accéder.

La mise en oeuvre d'un service se fait en accédant à un nœud de services MExE fournissant le service MExE. Le service est alors transféré vers le terminal concerné grâce à des mécanismes impliquant des protocoles de réseaux fixes ou mobiles ou sans fil, des liens Bluetooth, infrarouges ou sériels, des protocoles sans fil standardisés, des protocoles de l'Internet, etc. Ces nœuds de services peuvent être installés dans un contexte de commutation de circuits ou de paquets, d'Internet classique ou multimédia, etc. Cet environnement peut également contenir un serveur proxy chargé de traduire le contenu défini selon les protocoles standards de l'Internet en leur équivalent dans des systèmes dérivés optimisés pour les réseaux sans fil. Pour la souplesse du support des services MExE, le réseau sans fil fournit au terminal MExE un accès à une variété de services

support sur l'interface radio avec la possibilité de contrôler et de transférer une application d'un environnement de services MExE ailleurs. Une telle architecture (figure 2-3) devrait permettre de réaliser le concept du VHE (*Virtual Home Environment*) associé à la portabilité de l'environnement de service de l'abonné au-delà des limites de réseaux et des terminaux [GUYO_05].

Dans le cadre du GSM la prise en compte d'un pareil environnement est prévue à travers une table des services [GSM11.11]. Les spécifications du [GSM23.057] envisagent d'étendre ces services à la définition de profils d'utilisateur, à la négociation des services en fonction des capacités du terminal du client, à la découverte et à la gestion des services, à leur facturation, à leur qualité, à leur déploiement sous condition de satisfaction de certains critères de sécurité (intégrité des exécutables, permissions, usage de différentes clés publiques, etc.). Les principales exigences en rapport avec les services d'un pareil système sont consignées dans [GSM22.057].

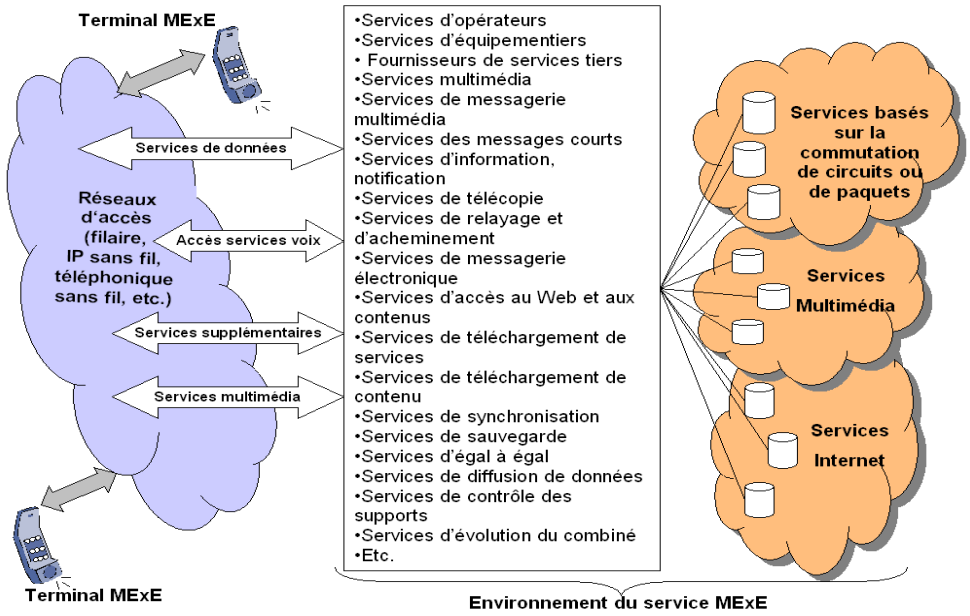


figure 2-3 : architecture générique de l'environnement MExE.

V. La SIM dans la famille des modules de sécurité

La carte SIM fait partie de la grande famille des modules de sécurité SAM (*Secure Access Module*) : des composants contenant des informations infalsifiables et ayant pour vocation de sécuriser l'accès à d'autres ressources. Connaître les cartes SAM dans leur architecture et leur fonctionnement, c'est donc également connaître en interne la SIM.

Une carte SAM est d'abord une carte à puce dont la description physique est détaillée dans [7816-3_97]. Son architecture habituelle (figure 2-4) renferme toujours une unité centrale de traitement, souvent un processeur cryptographique, et diverses sortes de mémoires : de la mémoire ROM pour conserver le code exécutable du système d'exploitation, de la mémoire RAM pour le stockage des instructions en cours d'exécution, et de la mémoire non volatile (mémoire EEPROM, FLASH, etc.) comme mémoire de masse. La sécurité de ce dispositif est assurée par une variété de défenses matérielles et logicielles gérées par le système d'exploitation, par l'entremise de l'unité de protection mémoire. Par exemple, le bus de données interne est chiffré afin d'éviter l'espionnage, ou

encore une protection des calculs cryptographiques est faite afin d'éviter des attaques par analyse de consommation d'énergie. En matière d'interfaces physiques, ces cartes à puce sont soit avec contact (conformité à la spécification ISO/IEC 7816) soit sans contact (conformité à la spécification ISO/IEC 14443), avec des commandes transportées sur une ligne sérielle. Récemment sont apparues d'autres interfaces de communication comme en premier lieu les interfaces USB avec un débit de 12 Mbit/s [TuCoSo05], et secundo les cartes mémoires MMC (*MultiMedia Card*).

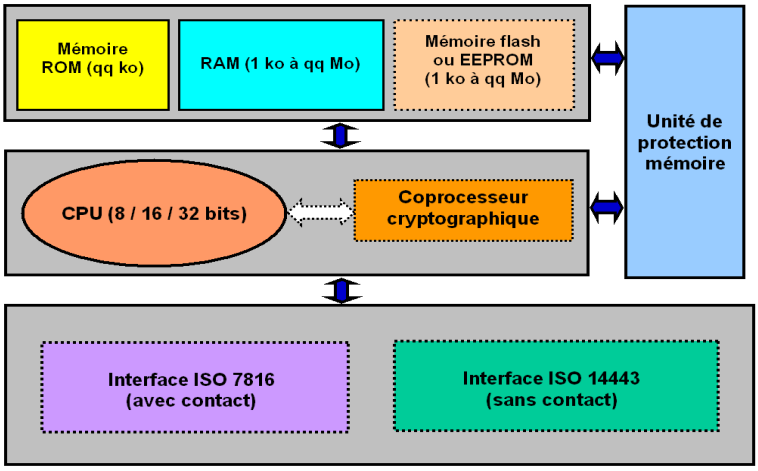


figure 2-4 : principaux blocs fonctionnels d'une carte SAM.

En matière de système d'exploitation, les cartes à puce supportent en général la technologie Java [SUN03] qui offre, d'une part, un contexte de développement d'applications utilisant un sous-ensemble du langage Java avec tous les atouts de la programmation orienté objet, et, d'autre part, un environnement sécurisé d'exécution.

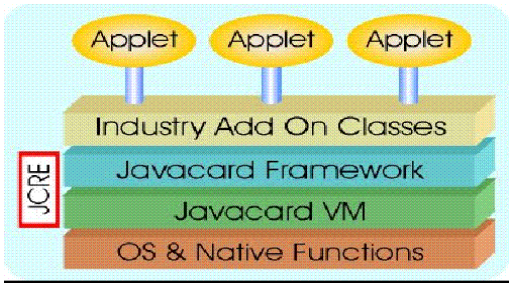


figure 2-5 : constituants habituels d'une carte Java.

L'environnement d'exécution Java (figure 2-5) est généralement constitué de cinq couches qui sont à partir de la plus basse : le système d'exploitation avec les fonctions de base, la couche correspondant à la machine virtuelle Java optimisée pour la carte à puce, la couche des fonctions standardisées pour l'environnement des cartes Java, la couche des classes additionnelles de fonctions spécifiques aux usages de la carte à puce, et enfin la couche des applications. La récente version 2.2.2 [SUN06] des spécifications de la carte Java supporte davantage de fonctions cryptographiques, de fonctions biométriques, d'interfaces de communication ; de plus elle se retrouve aussi bien sur des cartes d'identification sans contact que les nouvelles cartes SIM/USIM avec contact.

Le domaine de la téléphonie mobile est celui où les cartes SIM ont connu et continuent d'avoir un grand succès en jouant correctement leur rôle de SAM. Cela ne doit pas faire oublier le fait que le premier type d'utilisation des cartes à puce correspondait à la conservation des données relatives à son porteur. Cette utilisation a été importante dans les domaines des transactions bancaires avec les cartes bancaires B0, B0' et EMV, des transports, de la santé (carte Vitale en France), de l'identification des nationaux d'un pays avec les cartes d'identité [INES05] [PIV05] [BELPIC01] et passeports biométriques, du marquage au moyen des puces RFID (*Radio Frequence Identification*). La capacité de stockage des SAM étant limitée, il est de plus en plus envisagé de l'étendre au moyen des blobs (*binary blocs*) dont l'intégrité peut être contrôlée par des TPM (*Trusted Platform Module*) [TPM05].

Un autre type d'utilisation de la carte à puce a été d'en faire un nœud Internet [UR00] [UrTiLo02], soit par la présence en son sein d'une pile TCP/IP avec une adresse à part entière, soit en partageant la couche TCP/IP du terminal hôte. Si ces implémentations n'ont pas connu du succès dans le monde des réseaux IP du fait des capacités intrinsèquement limitées du support qu'est la carte à puce, nous pensons que la réalisation de la carte à puce EAP [UrLo03] constitue un pas vers une plus grande adoption de ce support dans le domaine des réseaux sans fil IP.

VI. La SIM et la sécurité de l'administration OTA

VI.1. Contexte général de sécurité

Schématiquement trois entités interviennent dans l'administration OTA comme nous le montre la figure 2-6 : le téléphone mobile, le réseau GSM et la passerelle OTA. Si nous faisons l'hypothèse que sont bien sécurisés le cœur du réseau GSM, la passerelle OTA et leur interconnexion, les autres lieux où la sécurité pourrait souffrir sont la carte SIM, son interconnexion avec l'équipement mobile, l'équipement mobile lui-même et l'interface radio entre le téléphone portable et le réseau.

La sécurité de la carte SIM est celle de la carte SAM décrite à la sous-section précédente. L'interface de la carte SIM avec l'équipement mobile est un contact physique ténu qui a lieu entre les deux entités [7816-1_98] [7816-2_99] ; cette frontière ne pose a priori pas de problème. L'équipement terminal héberge un système d'exploitation et des données dont les spécifications sont du domaine propriétaire. Comme cela s'est produit pour les ordinateurs personnels, leur corruption par des virus dans les années à venir est très probable. La mise à jour OTA qui concerne aujourd'hui essentiellement la carte SIM, devrait aussi concerner les terminaux mobiles dans le futur.

L'interface du téléphone mobile avec le réseau GSM est le lieu de réalisation des émissions/réceptions de l'utilisateur sur des fréquences radio. N'importe qui a la possibilité de les capter et de les manipuler. Il est donc nécessaire de prendre un minimum de précautions. La première est de ne jamais y faire circuler des clés, même chiffrées. Pour cette raison, le HLR et la carte SIM sont les seules dans l'architecture GSM à conserver la clé d'authentification *Ki* de l'abonné. Ainsi l'échange des clés, comme cela se passe lors de la mise à disposition du MSC/VLR par le HLR des clés de chiffrement *Kc*, demeure interne au réseau fixe de l'opérateur. S'agissant des autres informations qui devront passer par cette interface aérienne, il faudra principalement assurer leur confidentialité, leur intégrité, et les protéger contre le rejeu. Du fait de la mobilité de l'usager du GSM et du grand nombre d'abonnés de ce type de réseau, les problèmes d'identification,

d'authentification et de non répudiation devront impérativement être résolus lors de l'accès et de l'utilisation des services du réseau.

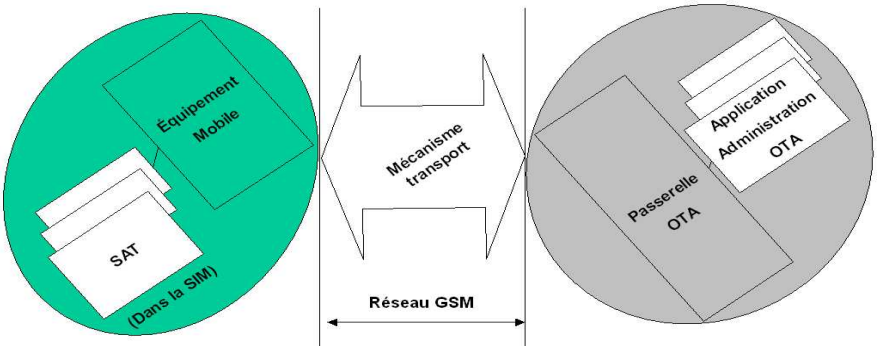


figure 2-6 : contexte général de sécurité dans l'administration OTA.

VI.2. Mécanismes de sécurité employés

Pour assurer la sécurité au niveau du GSM, des mécanismes sont décrits dans [GSM02.48] et [GSM03.48] ; ce sont : la procédure d'identification, celle d'authentification, la mise en place du chiffrement, les mécanismes de vérification d'intégrité des messages et de leur séquence, la détection du rejeu, et enfin la preuve de réception.

La procédure d'identification est employée par le réseau GSM pour demander au téléphone mobile de lui fournir des paramètres spécifiques d'identification comme par exemple le numéro IMSI (*International Mobile Subscriber Identity*) et le numéro IMEI (*International Mobile station Equipment Identity*). A partir de l'identité IMSI fournie, le réseau peut retrouver la clé d'authentification *Ki* qui lui correspond au niveau de la base de données AUC du réseau où a souscrit l'abonné. Cette clé est employée lors de l'authentification de la SIM par le réseau. L'identité IMEI quant à elle permet de vérifier que l'équipement mobile ne figure pas parmi une liste d'équipements dont l'accès au réseau doit être proscrit. Cette procédure d'identification va généralement précéder toute procédure d'authentification.

Lors du transfert d'un message, la passerelle OTA et l'équipement mobile sont les lieux d'application des mécanismes qui créent les conditions de sécurité par l'emploi d'un en-tête de sécurité [GSM03.48] accompagnant toujours le message à envoyer ou à recevoir. Cet en-tête contient des indications sur les paramètres de sécurité employés, les champs qui les abritent, la clé et l'algorithme de chiffrement employés, etc.

L'authentification peut se définir comme la vérification par une entité B que l'identité déclarée par une entité A est véritablement la sienne. Un premier degré d'authentification est l'authentification unilatérale procurant à l'acteur de l'authentification (le vérificateur), la preuve de l'identité du sujet de l'authentification (celui sur lequel porte la vérification). Du fait de la nature unidirectionnelle des mécanismes de transport par messages courts et USSD, ce type d'authentification est actuellement utilisé par le GSM ; c'est le réseau qui authentifie (vérifie l'identité de) l'entité qui veut communiquer avec lui. Il n'est pas impossible qu'avec l'apparition d'autres mécanismes de transport, le réseau n'ait aussi à s'authentifier auprès de la carte SIM ; dans ce cas on aurait un deuxième degré d'authentification, l'authentification mutuelle. En général, une authentification repose sur la présentation d'une combinaison de choses que l'on sait (mot de passe, code PIN, nom de connexion, adresse de courrier électronique, etc.), de choses que l'on possède (carte à puce,

badge, invitation, attestation, permis, etc.), et mieux encore de traits qu'on a (données de biométrie).

Le mécanisme associé à l'intégrité du message garantit qu'aucune corruption, accidentelle ou volontaire, du contenu du message n'a eu lieu au cours de son acheminement. Pour cela, les résultats des tests de redondance, des contrôles cryptographiques et des signatures numériques sont inclus dans l'en-tête de sécurité des paquets transférés entre l'équipement mobile et la passerelle OTA.

Le mécanisme de détection du rejeu fournit à l'entité recevant le message, le moyen de reconnaître qu'un même paquet sécurisé n'a pas déjà été reçu. La présence d'un compteur de 40 bits dans l'en-tête sécurisé permet de le réaliser. La protection de ce compteur est faite en l'incluant dans les champs dont l'intégrité est contrôlée. Parallèlement au rejeu, on peut aussi avec ce compteur garantir l'intégrité de la séquence (aucun changement intervenu dans l'ordre) des paquets sécurisés transmis.

La confidentialité du message assure que les messages échangés ne sont pas mis à la disposition ou divulgués à des entités non autorisées à en prendre connaissance. Elle est essentiellement réalisée au moyen d'algorithmes de chiffrement et de déchiffrement employant des clés symétriques périodiquement mises à jour. La cryptographie dans un contexte de clés symétriques nécessite le partage d'un secret ; on comprend alors l'emploi de la carte SIM pour conserver la clé, cachée à l'abonné, mais néanmoins manipulée par lui, sans risque de falsification. Le chiffrement est normalement appliqué à toutes les communications au sein du GSM, sauf celles concernant les messages courts en diffusion. Bien qu'un algorithme standard soit normalement employé, il est donné au mobile et/ou à l'infrastructure du réseau de l'opérateur de supporter plus d'un algorithme de chiffrement. En pareille situation, c'est l'infrastructure qui est responsable du choix de l'algorithme à employer (avec la possibilité de ne pas du tout chiffrer, et dans ce cas la confidentialité n'est pas assurée). Lorsque c'est nécessaire, le mobile peut indiquer au réseau les algorithmes de chiffrement qu'il supporte parmi un maximum de sept ; le réseau en sélectionne un sur la base d'un ordre de priorité défini à l'avance et le notifie au mobile. Le réseau ne fournit pas de service à un mobile indiquant qu'il ne peut supporter aucun des algorithmes de chiffrement requis par les spécifications [GSM 02.07]. Pour terminer, notons que s'agissant des messages de signalisation, la confidentialité ne s'applique qu'à des champs précis ; les éléments de signalisation inclus dans ces messages employés pour l'établissement de la connexion comme l'identificateur de protocole, la référence de la connexion, le type de message, l'identité du mobile (IMSI, TMSI ou IMEI selon les cas) ne sont pas protégés. Cependant, une fois la connexion établie, les éléments de signalisation relatifs à l'utilisateur comme l'IMEI, l'IMSI, le TMSI sont chiffrés [GSM02.09].

La preuve de réception indique à l'entité ayant expédié le message que l'entité destinataire a correctement reçu un paquet sécurisé d'une part, a réalisé les vérifications de sécurité nécessaires et a transmis son contenu à l'application destinataire finale d'autre part. Cette preuve peut être demandée à tout moment lors de l'envoi d'un paquet muni d'un en-tête sécurisé.

VI.3. La carte SIM au cœur de la sécurité

La carte SIM selon les spécifications du [GSM02.17] est l'entité abritant l'identité sans équivoque de l'abonné dénommée IMSI (*International Mobile Subscriber Identity*). La présence de la SIM dans un équipement mobile en bon état de fonctionnement est à l'origine de l'enregistrement du téléphone portable dans le réseau GSM. En effet, la

fonction première de la SIM, en conjonction avec le réseau, est de fournir et de participer à la vérification de l'identité IMSI ou TMSI (*Temporary Mobile Subscriber Identity*) d'un téléphone mobile accédant au réseau. De plus, elle procure à l'abonné le moyen d'être authentifié (par la réponse chiffrée au moyen de la clé *Ki* en son sein, au challenge émanant du réseau) et de conserver d'autres informations sur lui et les services auxquels il a souscrit. La carte SIM peut, en plus de l'application GSM, héberger d'autres applications. L'identification et l'authentification de l'utilisateur mises ensemble permettent d'effectuer un contrôle d'accès au réseau.

VI.3.1. La SIM, un espace de stockage sécurisé

La SIM est le lieu de stockage d'informations relatives à l'abonné [GSM11.11]. Ces informations peuvent être regroupées en trois catégories : les informations statiques mises en place durant la phase administrative (l'identité IMSI, la clé d'authentification *Ki* de l'abonné, le niveau de contrôle de son accessibilité, la table des services qu'elle supporte, les codes des fonctions à y exécuter/interpréter, la liste des canaux de fréquences radio à scruter périodiquement), les données mises en place de façon temporaire par le réseau (par exemple l'indicateur de la cellule géographique où se trouve le mobile à un instant donné, la clé de chiffrement des communications, le numéro de séquence de cette clé de chiffrement, les réseaux interdits, etc.), et les données relatives aux services supportés (par exemple la facturation, le choix du langage dans la communication homme/machine, etc.).

La SIM contient des informations permettant d'en vérifier le porteur, de manière à empêcher toute utilisation non autorisée. Elles sont désignées par CHV (*Card Holder Verification information*) et sont au nombre de deux : CHV1 et CHV2 connus anciennement sous le nom de code PIN (*Personal Identity Number*). Un certain nombre d'échecs de fourniture de la valeur du CHV1 déclenche le blocage de la SIM. Seule la communication de la bonne valeur du CHV2 permet de la débloquer et de remettre à zéro le compteur d'erreurs de fourniture associé à CHV1. Après un certain nombre d'essais infructueux de déblocage par CHV2, la SIM se voit irrémédiablement bloquée.

Les données présentes dans la SIM sont organisées [GSM11.11] selon une arborescence dont la racine est nommée MF (*Master File*). On y trouve des répertoires appelés DF (*Dedicated File*) et des fichiers classiques dénommés EF (*Elementary File*). La figure 2-7 illustre l'organisation logique de l'espace de la SIM. En gris apparaissent certains fichiers importants comme celui abritant la table des services de la SIM (EF_{SST}), le fichier des messages courts reçus (EF_{SMS}), le fichier du langage préféré de l'abonné (EF_{LP}), le fichier contenant le numéro d'identification unique et secret de l'utilisateur (EF_{IMSI}), le fichier contenant le numéro d'identification unique de la SIM (EF_{ICCID}), le fichier contenant le ou les numéros de téléphone par lesquels l'abonné peut être appelé (EF_{MSISDN}), etc. Le répertoire DF_{GSM} renferme les fichiers liés à la mobilité, alors que le $DF_{TELECOM}$ héberge les paramètres couramment employés en téléphonie.

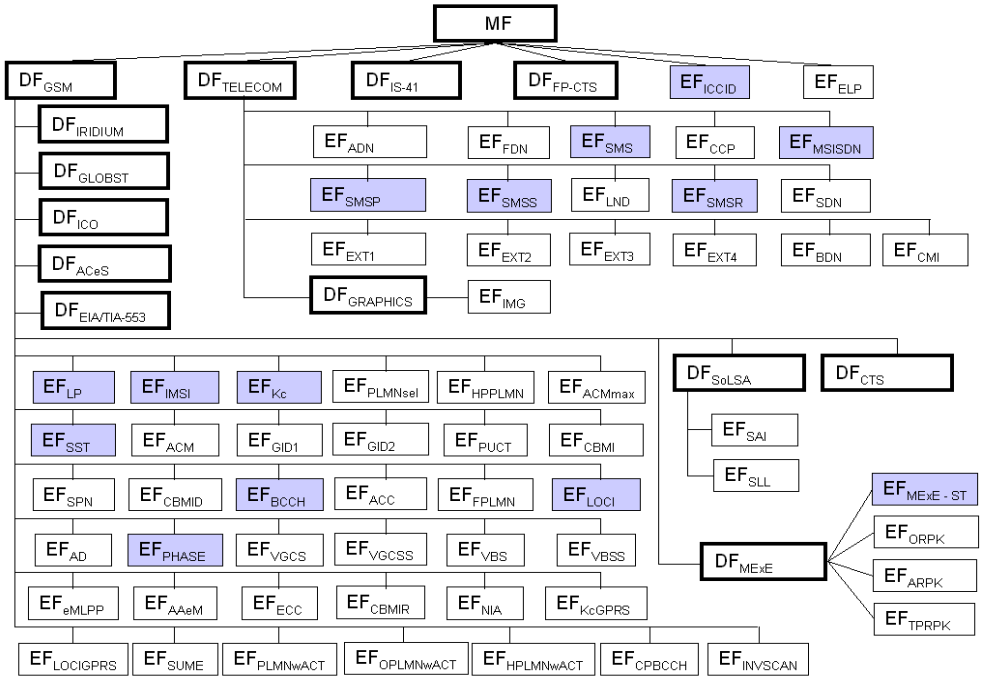


figure 2-7 : organisation des fichiers et répertoires GSM dans la SIM.

Pour des raisons d'anonymat et de confidentialité, le numéro IMSI est très rarement envoyé en clair sur l'interface radio. Après l'authentification réussie de la SIM, le sous-système de base BSS calcule un numéro temporaire dénommé TMSI (*Temporary Mobile Subscriber Identity*) et l'envoie au mobile qui s'en sert dorénavant comme identité locale. Cette identité temporaire est conservée avec d'autres informations dans le fichier classique EF_{LOCI} de la SIM. Les formats des identités et numéros employés au niveau du GSM sont tous rassemblés dans [GSM03.03].

Chaque fichier élémentaire de la SIM peut supporter l'une des opérations suivantes : lecture (READ ou SEEK), modification (UPDATE), invalidation (INVALIDATE), réhabilitation (REHABILITATE). Les opérations d'invalidation de fichiers ont pour effet de les rendre inutilisables, et par voie de conséquence paralysent les applications employant les fichiers ainsi traités. En général un traitement sur un fichier n'est possible que si l'une des cinq conditions d'accès est satisfaite ; elles peuvent être : ALWAYS (aucune restriction au traitement), CHV1 (permission pour le traitement si le code PIN1 a déjà été fourni correctement ou si sa vérification a été désactivée), CHV2 (permission pour le traitement si la valeur correspondant au code PIN2 a déjà correctement été fournie ou si ce contrôle a déjà été désactivé), ADM (permission de traitement accordée seulement à l'autorité administrant la puce), NEVER (interdiction de réaliser le traitement via l'interaction SIM/ME). Un service qui démarre ne peut fonctionner que si les conditions d'accès des fichiers employés sont vérifiées par le statut de celui qui le demande. Ainsi pour le fichier EF_{SST} associé à la table des services de la SIM, on trouve comme conditions d'accès "READ=CHV1, UPDATE=ADM, INVALIDATE=ADM, REHABILITATE=ADM", signifiant que sa lecture n'est possible qu'après fourniture du code PIN, et que les autres opérations ne sont autorisées que lorsque l'on dispose des droits de l'administrateur.

Chaque répertoire ou fichier élémentaire doit, avant son utilisation, être défini comme objet en cours d'utilisation au moyen de la commande SELECT FILE. On ne peut référencer un fichier élémentaire que dans la mesure où le répertoire qui le contient est défini comme répertoire courant.

VI.3.2. La SIM, un environnement de traitement sécurisé

Les fonctions cryptographiques spécifiques au GSM et réalisées par la SIM sont définies dans [GSM03.20]. Elles sont au nombre de trois : A3, A8 et A5. Parfois en remplacement des fonctions A3 et A8 on trouve la fonction A38.

Les deux premières fonctions ou leur remplaçant sont employées lors de l'authentification de la SIM par le réseau GSM. Leur utilisation est illustrée à la figure 2-8 associée à la procédure d'authentification unilatérale de la SIM par le réseau qui en est l'initiateur ; en même temps que le réseau authentifie, les conditions de chiffrement/déchiffrement sont mises en place de part et d'autre. Chacune des fonctions utilise, en entrée, la valeur aléatoire *RAND* de 128 bits reçue nouvellement du réseau et la clé d'authentification de l'abonné dénommée *Ki*, également de 128 bits et conservée dans la SIM. Le résultat de la fonction A3 dénommé SRES (*Signed RESponse*) est sur 32 bits et correspond à la signature du *RAND* par le mobile. La fonction A8 permet de calculer la clé de chiffrement/déchiffrement *Kc* de 64 bits. La valeur SRES est renvoyée au réseau pour finaliser la procédure d'authentification, alors que la clé *Kc* est conservée localement et servira au chiffrement et au déchiffrement des futurs messages échangés entre le mobile et le réseau.

La dernière fonction cryptographique spécifique, A5, permet de chiffrer et de déchiffrer les messages échangés entre le mobile et le réseau (données de l'utilisateur et signalisation réseau), une fois que la clé de chiffrement a été déduite de part et d'autre. Elle emploie en entrée le numéro de la trame courante sur 22 bits et la clé *Kc* de 64 bits pour fournir en sortie deux blocs de 114 bits : l'un de chiffrement et l'autre de déchiffrement. Ces blocs, employés dans des opérations de "OU-exclusif bit à bit", permettent de chiffrer/déchiffrer les blocs de 114 bits du message en entrée. Pour que la fonction marche, il est nécessaire d'avoir une synchronisation entre le réseau et le mobile, en ce sens que le bloc servant d'un côté pour le chiffrement doit servir de l'autre côté pour le déchiffrement et inversement. Les traitements de chiffrement se déroulent juste avant la modulation.

Dans le cadre de la réalisation des services de messages courts, le standard GSM 03.48 [GSM03.48] utilise un format hautement sécurisé pour l'acheminement des commandes et des codes exécutables à destination du téléphone mobile. Dans la partie en-tête de tels messages, se retrouvent des informations concernant premièrement la clé et l'identificateur de l'algorithme de chiffrement/déchiffrement employés, deuxièmement la clé et l'identificateur de l'algorithme employés pour garantir l'intégrité, et troisièmement le code d'intégrité du message (champs RC pour *Redundancy Check-*, CC pour *Cryptographic Checksum,* ou DS pour *Digital Signature*).

Pour la réalisation de tous les traitements spécifiques énumérés précédemment, la carte SIM s'appuie dans la majeure partie des cas sur les services assurés grâce aux API cryptographiques Java disponibles. Ainsi on retrouve habituellement sur des cartes SIM disposant de l'environnement JC2.1, des algorithmes cryptographiques tels que SHA1 [SHA1_02], DES et 3DES [DES99], MD5 [MD5_92] et RNG (*Random Number Generator*). L'absence de certains algorithmes cryptographiques en natif sera à l'origine de développements de codes additionnels ; vont se retrouver dans cette situation par exemple les fonctions de chiffrement RC4, de génération de nombres pseudo aléatoires PRF, de calcul d'empreintes chiffrées HMAC-MD5 et HMAC-SHA1 [HMAC97].

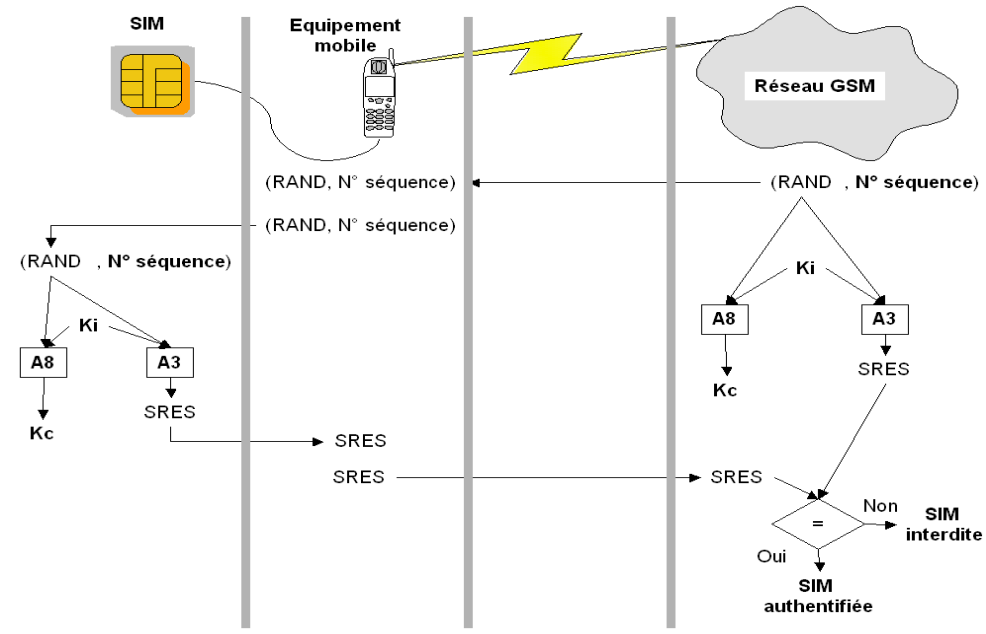


figure 2-8 : déroulement de l'authentification de la SIM par le réseau.

Enfin, notons qu'il est de plus en plus possible à de nouvelles applications d'être hébergées sur la SIM, faisant d'elle une carte multi application rassemblant en son sein l'équivalent des nombreuses cartes qui remplissent actuellement les porte-feuilles. C'est la présence d'environnements MExE (clés publiques des opérateurs, clés publiques des fournisseurs de services, etc.) dans la carte SIM qui rend cette évolution possible aujourd'hui.

VI.3.3. La SIM, un environnement administrable à distance

L'idée fondamentale d'administration à distance des cartes SIM est de pouvoir faire exécuter en son sein une ou plusieurs commandes qui lui sont envoyées en empruntant l'un des types de transport qu'offre le réseau GSM.

Dans la gestion à distance d'une carte à puce SIM on peut distinguer deux principales catégories d'opérations : celles en rapport avec l'accès au système de gestion de fichiers de la SIM, et celles relatives à la gestion des applications (*applets*) qu'elle abrite. Les opérations sur les fichiers vont faire appel aux commandes telles que SELECT FILE, READ, WRITE, etc.. évoquées dans la section précédente. La gestion des applications comprend la possibilité de les charger, de les installer, de les désinstaller. Aussi, cinq commandes importantes sont nécessaires : LOAD (pour le transfert du code d'une application dans un domaine de sécurité de la carte SIM), INSTALL (prise en compte dans le registre des applications accessibles sous le contrôle d'un domaine de sécurité), DELETE (suppression de l'application du domaine de sécurité et des registres contenant une référence la concernant), PUT KEY (pour mettre en place ou modifier des clés de sécurité relatives aux domaines de sécurité ou même aux applications) et SET STATUS (pour modifier l'état dans le cycle de vie d'une carte ou d'une application). Ces fonctionnalités sont plus détaillées dans les spécifications de l'organisation GlobalPlatform [GLOBPL03].

Avant de terminer, faisons remarquer qu'une session d'administration à distance s'effectue en trois étapes :

- l'ouverture d'un canal sécurisé entre le domaine de sécurité de la carte SIM ou parfois même une application spécifique d'un domaine de sécurité de la carte SIM

et une entité extérieure à la carte SIM. C'est généralement à la suite d'une authentification mutuelle réussie que ce canal sécurisé est ouvert. Par exemple le protocole SCP01 (*Secure Channel Protocol '01'*) défini dans [GLOBPL03] emploie au cours de l'authentification d'une part deux clés cryptographiques statiques S-ENC (pour le calcul de la clé de chiffrement de la future session) et S-MAC (pour le calcul de la clé associée au code d'intégrité des messages de la future session), et d'autre part des valeurs du challenge et de la réponse au challenge pour dériver les clés de session S-ENC et S-MAC qui serviront après le succès de l'authentification ;

- l'emploi du canal sécurisé pour réaliser l'opération d'administration. Les clés de session S-ENC et S-MAC issues de l'authentification qui vient d'avoir lieu sont employées. Le protocole SCP01 prévoit même une clé DEK (*Data Encryption Key*) pour chiffrer les données sensibles à transférer au cours de la phase effective d'administration, comme ce pourrait être le cas de clés privées RSA [RSA78] ;
- la fin de l'opération d'administration intervenant sur décision d'une des parties ou lors de la survenue d'une erreur au moment de l'authentification.

C'est vers une architecture devant permettre un tel fonctionnement dans l'environnement des réseaux sans fil IP que nos efforts vont se porter.

VII. Conclusion

Dans ce chapitre, nous avons présenté dans un premier temps les deux principales entités fonctionnelles d'un téléphone mobile, et les mécanismes internes qu'elles emploient afin d'assurer les services de base, les services supplémentaires standardisés mais aussi les services supplémentaires qui se développeront dans le futur, afin de rendre ces services accessibles de partout avec des téléphones portables préparés en conséquence. Dans un deuxième temps, nous nous sommes focalisés sur la SIM en tant que dispositif de la classe des SAM, s'inscrivant dans le modèle de sécurité du GSM comme cadre sécurisé de conservation d'informations sensibles mais aussi comme contexte de traitement permettant de mettre en œuvre les mécanismes de sécurité indispensables à l'administration OTA.

Si les premières utilisations de la carte à puce au sein des réseaux IP dans l'objectif d'en faire des nœuds Internet n'ont pas été un franc succès, la carte à puce EAP, quant à elle, constitue à nos yeux une voie offrant des perspectives beaucoup plus intéressantes ; c'est l'objet des prochains chapitres.

Chapitre 3 : Les réseaux sans fil IP

Comme nous nous intéressons à une architecture destinée aux réseaux sans fil IP, après avoir traité de l'administration OTA dans le réseau GSM, ce chapitre est consacré à la présentation des réseaux sans fil. D'abord les traits importants des spécifications 802.11 introduisant le premier réseau local sans fil seront présentés. Puis l'architecture 802.1X proposée dans le but de permettre la mise en place d'un cadre sécurisé d'échanges dans les réseaux locaux sans fil sera traitée. La présentation de l'architecture 802.11i dans la troisième partie montrera ce qui constitue actuellement le sommet de la sécurité dans les réseaux locaux sans fil IP. Les technologies de réseaux sans fil IP métropolitains comme *WiMax* et *WiMax mobile* sont aussi en train de se développer dans la perspective d'offrir des accès Internet haut débit partout ; la quatrième partie de ce chapitre y est consacrée. La fin du chapitre évoque le rôle que la carte à puce pourrait jouer dans ces réseaux sans fil IP.

I. Généralités sur la famille des réseaux locaux 802.11 (WLAN)

Ratifié en 1999 par le Comité 802 de l'IEEE, le standard 802.11 est à la base du développement des réseaux sans fil IP connu sous l'appellation *Wireless Local Access Network* (WLAN). Avec l'amélioration des performances des couches MAC (*Medium Access Control*) et physique (PHY), des débits importants ont pu être atteints, concurrençant les réseaux Ethernet 802.3 filaires classiques.

tableau 3-1 : quelques caractéristiques des principaux réseaux sans fil IP.

Standards	Catégories	Applications	Débits théoriques	Portées maximales	Fréquences ou longueurs d'onde
802.11	WLAN	Réseau local	1-2 Mbit/s	30 m	2,4 GHz
802.11a	WLAN	Réseau local	54 Mbit/s	30 m	5 GHz
802.11b (WiFi)	WLAN	Réseau local	11 Mbit/s	100 m	2,4 GHz
802.11b+ (Turbo)	WLAN	Réseau local	22 Mbit/s	100 m	2,4 GHz
802.11g	WLAN	Réseau local	54 Mbit/s	70 m	2,4 GHz
802.15	WPAN	PC ↔ PDA PC ↔ téléphone	1 ou 12 Mbit/s	30 m ou 10 m	2,4 GHz
IrDa	WPAN	PC ↔ périphérique	4 Mbit/s	1 m	850 nm
802.16	WMAN	Réseau de campus fixe	120 Mbit/s	50 km	10 – 66 GHz
802.16e	WMAN	Réseau de campus mobile	120 Mbit/s	5 – 15 km	10 – 66 GHz

En effet, si dès 1999 un débit maximum de 2 Mbit/s pouvait être atteint avec le standard 802.11, la venue du 802.11b labellisé WiFi (tableau 3-1) a contribué à l'expansion des réseaux sans fil. En effet le débit maximal théorique du WiFi était de 11 Mbit/s, donc supérieur à ce que pouvait fournir les réseaux Ethernet habituels, avec en plus des avantages comme le faible coût de réalisation, la simplicité de déploiement et d'extension, le gain en terme de mobilité (pas d'assujettissement à la position d'une prise réseau), l'interopérabilité des équipements certifiés WiFi, etc. Avec le standard 802.11a

[802.11a_99], et surtout le 802.11g [802.11g_03] utilisant la même bande de fréquences que le WiFi, des débits théoriques de 54 Mbit/s ont été annoncés, permettant d'espérer sur le médium aérien près de la moitié du débit connu en Fast Ethernet sur les câbles. Des travaux sont en cours au niveau du *Task Group N* de l'IEEE sur le successeur prévu pour le 802.11g et nommé 802.11n dont le débit maximum offert devrait atteindre 500 Mbit/s.

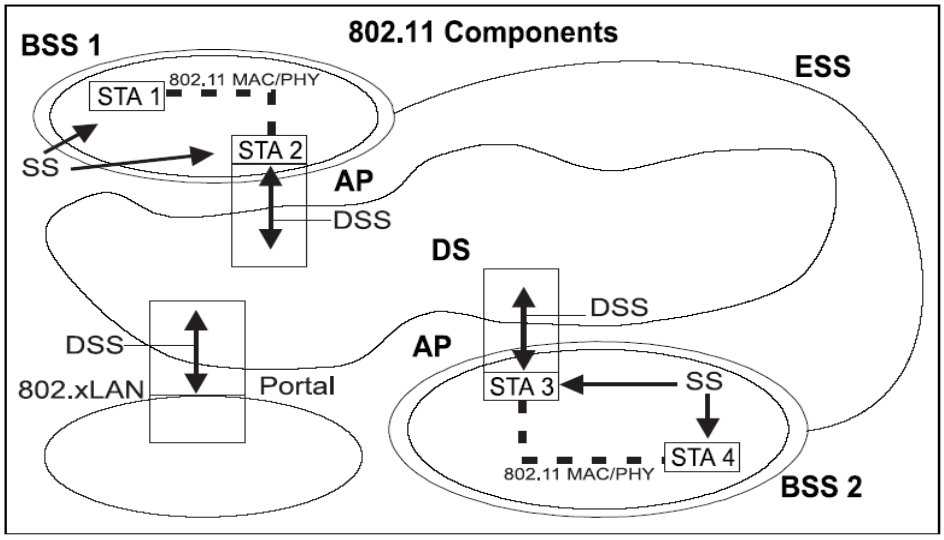


figure 3-1 : constituants d'un réseau sans fil IP 802.11.

A la figure 3-1 sont représentés les éléments constituant les différentes architectures possibles de réseaux sans fil 802.11. D'abord, on distingue les stations dénommées STA qui sont les entités équipées de moyens radio leur permettant de communiquer entre elles dans une certaine proximité géographique ; elles forment la structure de réseau de base appelée BSS (*Basic Service Set*). Un BSS qui n'est relié à aucun autre réseau est appelé IBSS (*Independant BSS*) et fonctionne en *mode ad hoc*. Un BSS peut être raccordé à un réseau local dénommé DS (*Distributed System*) grâce à un équipement particulier appelé AP (*Access Point*), et de ce fait permettre aux stations qui composent ce BSS d'accéder aux ressources disponibles sur le réseau local raccordé, et inversement : on dit que ce réseau sans fil fonctionne en *mode architecture*. Plusieurs BSS raccordés via des DS créent de façon logique un ESS (*Extended Service Set network*) qui permet aux stations composant les BSS en jeu de communiquer entre elles.

Dans le cadre de la sécurisation des réseaux 802.11, deux types de services de sécurité sont prévus pour être fournis aussi bien par les stations que par les points d'accès ; ce sont : les services d'authentification et les services de chiffrement.

Le service d'authentification est assuré sur la base d'un partage de secret entre deux stations. L'une d'elles est le point d'accès qui fournit des services aux stations et au système de distribution. Le service d'authentification suppose la présence du service de chiffrement qui, au cours de l'authentification, va être employé pour chiffrer le contenu d'un envoi sur le médium aérien. Par défaut, un réseau 802.11 est comme un système ouvert puisque aucune authentification n'y est faite et ainsi n'importe quelle station peut s'insérer dans le réseau.

Le service de chiffrement est symétrique et proposé par le biais de l'algorithme WEP (*Wired Equivalent Privacy*) qui lui-même se fonde sur l'algorithme RC4. Il emploie en

entrée une clé de 40 bits de longueur, un vecteur d'initialisation de 24 bits et un contrôle d'intégrité CRC-32.

Malheureusement, comme ce réseau sans fil emploie plusieurs des principes de base des réseaux locaux filaires, de nouvelles menaces à la sécurité sont apparues, à la faveur des faiblesses du WEP, comme les attaques parfois qualifiées d'attaques à partir d'un parking, puisque l'on est incapable de restreindre à un groupe autorisé l'accès au médium hertzien comme c'est le cas au niveau des réseaux filaires. Les principales faiblesses reconnues au réseau 802.11 sont à nos jours :

- le mauvais fonctionnement de l'authentification puisqu'il est possible d'enregistrer une clé et de l'employer à nouveau dans le futur [BoGoWa01] ;
- la faiblesse du mécanisme de signature des paquets (chiffrement du CRC) puisqu'il est possible de modifier un octet dans le paquet chiffré et de continuer à calculer une valeur correcte de CRC [BoGoWa01] ;
- la difficulté à déployer à grande échelle l'architecture de confidentialité des informations puisque ce sont des valeurs statiques de 40 à 104 bits qui sont partagées entre le point d'accès et les stations sans fil. De plus les secrets partagés peuvent être trouvés en enregistrant un million de trames chiffrées [FluMaSh01].

Afin de remédier à ces imperfections et de donner aux utilisateurs la garantie de la confidentialité, de nouvelles architectures de sécurité ont vu le jour : la norme 802.1X suivie des spécifications du WPA (*WiFi Protected Access*), et finalement le standard 802.11i (ou WPA v2).

II. Architecture de sécurité 802.1X

Les spécifications 802.1X [802.1X_01] dérivent du standard 802.11b et ont pour vocation de sécuriser les réseaux sans fil IP 802.11b, 802.11a et 802.11g en instaurant un mécanisme physique de protection sur la base d'un contrôle des ports d'accès au réseau.

Dans un environnement 802.1X on distingue trois principales entités comme représentées à la figure 3-2 :

- le *supplicant*, un dispositif cherchant à accéder à des ressources d'un réseau sans fil, un peu comme la station dans le 802.11. On l'appellera dans la suite de notre document le demandeur d'accès ;
- l'authentifiant (*authenticator*) ou vérificateur/contrôleur d'accès qui est typiquement un point d'accès au réseau sans fil. Il est chargé d'appliquer la politique de sécurité d'accès au réseau décidée par une infrastructure AAA (*Authentication, Authorization and Accounting*) en fonction des lettres de crédits que lui présente le demandeur d'accès ;
- le serveur d'authentification (*authentication server*) qui est relié à une infrastructure AAA conservant d'habitude les profils des utilisateurs (abonnements, lettres de crédits, etc.) et leurs paramètres d'authentification (clés cryptographiques symétriques, certificats X.509, etc.). Généralement la plupart des méthodes d'authentification susceptibles d'être employées par les demandeurs d'accès sont centralisées sur le serveur d'authentification, évitant la multiplication des mises à jour qui se produiraient si elles étaient faites au niveau des authentifiants.

Au terme du processus d'authentification réussie, le port contrôlé de l'authentifiant va permettre au demandeur d'accès de disposer des services que le réseau l'autorise à

employer. Autrement, ce port contrôlé reste dans un état "non autorisé" empêchant le demandeur d'accéder aux services du réseau.

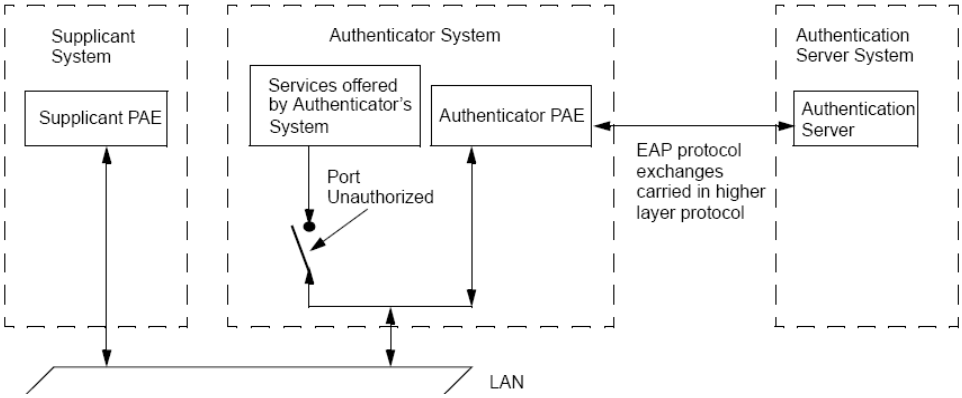


figure 3-2 : principaux acteurs de l'architecture de sécurité 802.1X.

Le standard 802.1X se singularise par le fait qu'il introduit l'emploi d'un protocole nommé EAP (*Extensible Authentication Protocol*) [EAP04] qui va servir de moyen de transport d'un grand nombre d'autres protocoles réalisant des mécanismes ou scénarii d'authentification. Pour que cela se passe bien, ce standard a défini, entre le demandeur d'accès et l'authentifiant, un protocole chargé d'encapsuler EAP : le protocole EAPoL (*EAP over LANs*) sensé convenir à tout LAN de type Ethernet employé entre le demandeur d'accès et l'authentifiant.

Cinq sortes de trames sont définies dans le protocole EAPoL : *EAP-Packet* pour le transport d'un paquet EAP, *EAPOL-Start* pour la demande d'ouverture de session EAPoL, *EAPOL-Logoff* pour la clôture d'une session EAPoL, *EAPOL-Key* pour le transport d'une clé, *EAPOL-Encapsulated-ASF-Alert* pour le transport de messages de gestion du genre des alertes SNMP (*Simple Network Management Protocol*) [SNMP02].

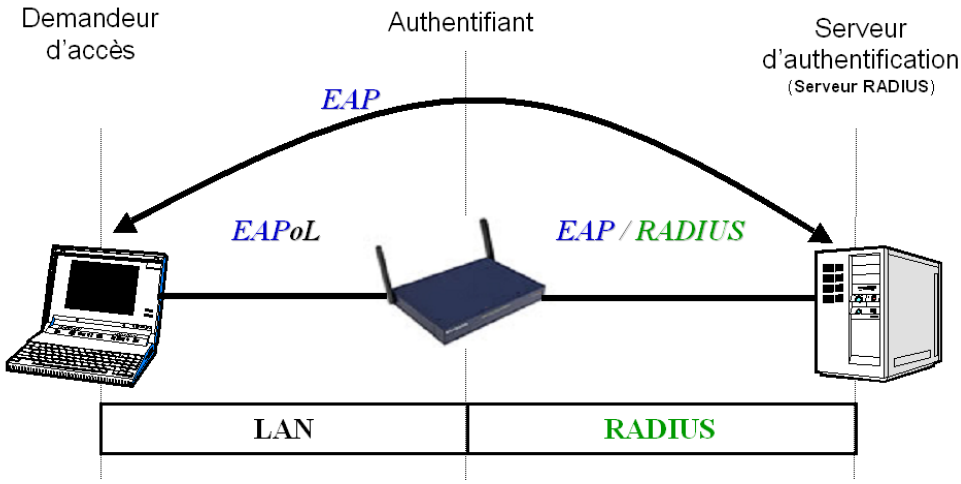


figure 3-3 : protocoles dans l'environnement 802.1X.

Entre l'authentifiant et le serveur d'authentification, le standard 802.1X ne définit rien de particulier. Les protocoles comme RADIUS (*Remote Authentication Dial In User*

Service) [RFC2865] [RFC3579] et DIAMETER [DIAM03] conviennent généralement au transport d'EAP sur ce segment du réseau. Le standard donne néanmoins la possibilité d'héberger sur la machine associée à l'authentifiant, le serveur d'authentification ; en pareil cas il n'est pas nécessaire qu'il y ait RADIUS entre les deux entités pour communiquer.

La figure 3-3 illustre des liens protocolaires existant entre les principaux acteurs présents dans un environnement de réseau 802.1X.

III. Architecture de sécurité WPA

Le WPA (*WiFi Protected Access*) est un ensemble de spécifications faites par la *WiFi Alliance*. Elles correspondent aux spécifications du 802.11 revêtues de certains aspects de sécurité du 802.11i. A mi-chemin entre les normes 802.11 et le 802.11i, elles constituent un progrès remarquable dans l'amélioration du niveau de sécurité (protection des données au moyen du chiffrement, contrôle d'accès au réseau grâce à une authentification) des réseaux locaux sans fil IP certifiés WiFi depuis que se sont révélées les failles du WEP. Ces améliorations du niveau de sécurité ont été obtenues simplement en modifiant le logiciel des équipements certifiés WiFi (près de 650 à l'époque). Ceci a permis de contrer toutes les attaques du WEP, et aussi d'offrir une possibilité d'authentification robuste. Le WPA fonctionne soit selon le standard 802.1X pour les entreprises (usage d'un serveur d'authentification RADIUS), soit en mode de partage de clé (mot de passe ou clé secrète seulement nécessaires, pas besoin de serveur d'authentification) dans les contextes SOHO (*Small Office / Home Office*).

WPA combat les vulnérabilités du WEP en utilisant :

- un vecteur d'initialisation de 48 bits au lieu de 24 bits, avec des règles définies pour son choix et sa vérification ;
- un MIC (*Message Integrity Code* appelé Michael) de 64 bits comme signature robuste de chaque trame en remplacement du CRC-32 ;
- la génération et la distribution de clés à partir de nombres aléatoires échangés au départ (lutte contre les attaques de type "homme du milieu"), ainsi que l'emploi du protocole TKIP (*Temporal Key Integrity Protocol*).

Le protocole TKIP est capable de générer un grand nombre de clés à partir d'une clé racine habituellement appelée clé PMK (*Pairwise Master Key*). Cette clé est sensée correspondre soit à une clé PSK (*Pre-Shared Key*) partagée à l'avance entre le demandeur d'accès et le point d'accès, soit à la clé mise à disposition du point d'accès par le serveur d'authentification à la suite d'une authentification mutuelle réussie.

A partir d'une clé PMK, le protocole TKIP dérive une clé PTK (*Pairwise Temporal Key*) de 512 bits en s'appuyant sur une fonction PRF (*Pseudo Random Function*) de génération de nombres pseudo-aléatoires (figure 3-4). La clé PTK est ensuite éclatée en trois parties : un paquet de 128 bits employé lors de la "phase de 1" de la fonction de mixage pour le chiffrement RC4, et deux paquets de 64 bits. Les deux participants à l'échange sur le médium aérien emploieront chacun l'une de ces clés de 64 bits pour servir lors des calculs de MIC (*Message Integrity Code*).

En somme, les clés dérivées par TKIP vont être employées pour la fourniture des services de confidentialité, d'intégrité et d'authenticité des messages échangés sur le médium aérien.

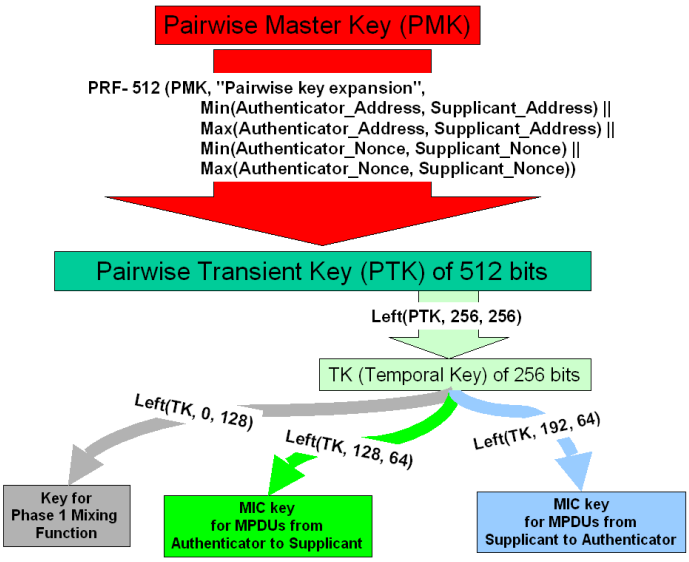


figure 3-4 : hiérarchie des clés avec le protocole TKIP.

IV. Architecture de sécurité 802.11i

La norme 802.11i [802.11i_04] est dérivée des spécifications du 802.11b. Elle vient renforcer l'architecture 802.1X grâce à une amélioration des mécanismes d'authentification à l'entrée des réseaux sans fil IP. Elle est totalement reprise par la *WiFi Alliance* à travers le WPAv2.

Dans la norme 802.11i est introduite la notion importante de réseau fortement sécurisé RSN (*Robust Security Network*). Dans un RSN toute association entre deux entités est le résultat d'une authentification mutuelle, soit entre deux stations dans le cadre d'un IBSS (*Independant Basic Service Set*), soit entre une station et un point d'accès, ou encore entre un point d'accès et un serveur d'authentification dans le contexte d'un ESS (*Extended Service Set network*).

Le standard 802.11i permet de fabriquer un ensemble de clés bien hiérarchisées ayant pour racine la PMK (*Pairwise Master Key).* Ceci est réalisé à partir d'une clé PSK partagée à l'avance entre le point d'accès et un demandeur d'accès, ou bien à partir d'une clé principale MK (*Master Key*) obtenue à la fin d'un scénario d'authentification EAP réussie entre un demandeur d'accès et le serveur d'authentification qui la met à la disposition du point d'accès. Ces clés vont, en combinaison avec les algorithmes relevant des protocoles CCMP (*Counter mode with CBC-MAC Protocol*) et optionnellement TKIP (*Temporal Key Integrity Protocol*), permettre de garantir la confidentialité et l'intégrité des messages échangés sur le médium aérien. La figure 3-5 illustre des échanges entre un demandeur d'accès et un point d'accès. Il en résulte la production d'un ensemble de clés dont l'usage va contribuer à sécuriser les communications sur le réseau sans fil.

La figure 3-6 présente la hiérarchie et le mode de génération des clés que le point d'accès et le demandeur d'accès peuvent ensuite produire afin de sécuriser leurs échanges.

Toutes ces clés s'emploient dans un contexte de chiffrement symétrique des échanges ayant cours sur le médium aérien entre un demandeur d'accès et un point d'accès. Ainsi, les clés temporaires TK (*Temporal Key*) servent au chiffrement et au déchiffrement des données utiles transportées. Les clés KEK (*EAPOL-Key Encryption Key*) sont attachées à la confidentialité du champ réservé au transport d'une clé. Les clés KCK

(*EAPOL-Key Confirmation Key*) vont servir dans le calcul d'une signature numérique MIC (*Message Integrity Code*) assurant l'intégrité d'une trame transportant une clé.

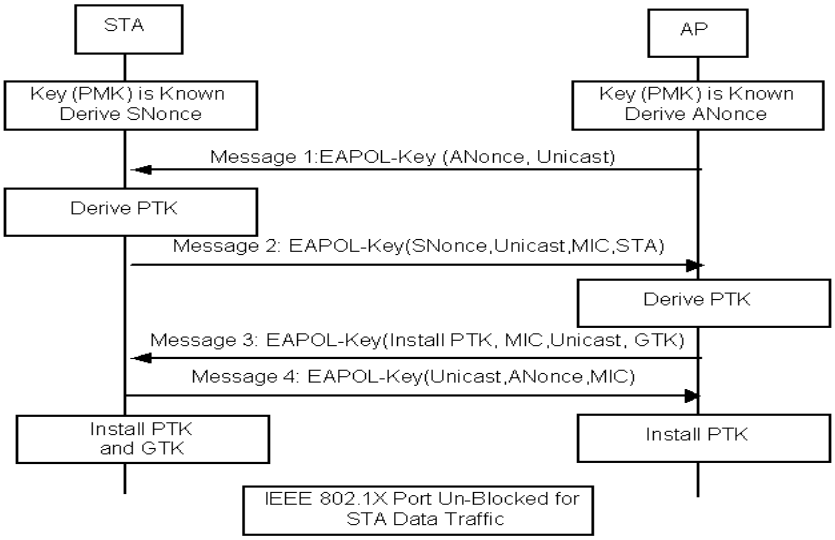


figure 3-5 : distribution de clés entre point d'accès et demandeur d'accès.

Le standard 802.11i prévoit aussi la production d'une autre hiérarchie de clés dite GTK (*Group Temporal Key*) destinées à un usage de groupe (fermé ou ouvert) de demandeurs d'accès (services multicast).

Le protocole CCMP fait usage de l'algorithme de chiffrement AES (*Advanced Encryption Algorithm*) [NIST01] avec toujours des clés et des blocs de 128 bits chacun. CCMP fonctionne en mode CCM [CCM03] combinant à son tour deux autres modes : CTR (*Counter mode*) pour le chiffrement/déchiffrement, et CBC-MAC (*Cipher Block Chaining Message Authentication Code*) pour garantir l'intégrité de la charge utile et de certains champs de l'en-tête de la trame. CCM utilise à chaque session une nouvelle clé TK combinée à un germe correspondant au numéro du paquet sur 48 bits.

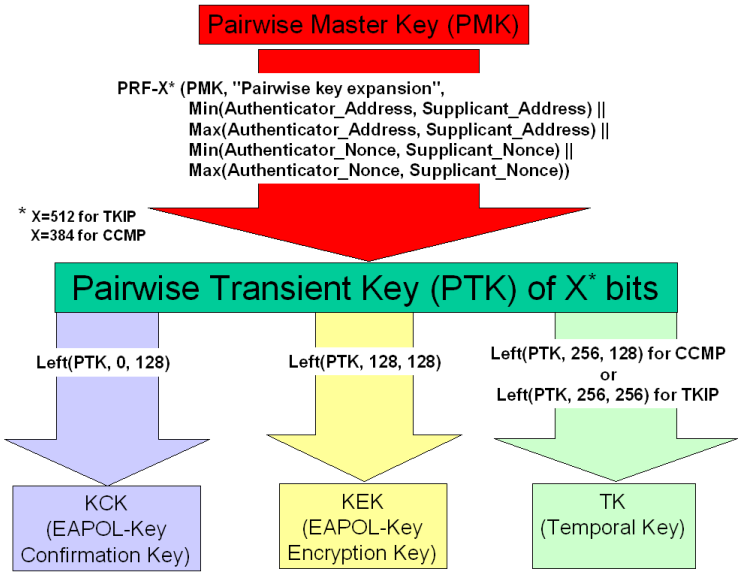


figure 3-6 : hiérarchie des clés à partir de la clé PMK.

Au tableau 3-2 nous présentons un récapitulatif des principales caractéristiques des protocoles de sécurité que sont le WEP, le WPA avec TKIP et le WPAv2 avec CCMP : algorithme de chiffrement, taille des clés, mode production et de gestion des clés, protections assurées.

tableau 3-2 : caractéristiques des protocoles de sécurité dans les WLAN.

	WEP	WPA/TKIP	WPA2/CCMP
Algorithme de chiffrement	RC4	RC4	AES
Taille des clés	40 bits	256 bits dont 128 bits pour le chiffrement, et 64 bits pour l'authentification	128 bits
Taille du vecteur d'initialisation	24 bits	48 bits	48 bits
Mode d'obtention de la clé du paquet	Concaténation	Fonction de mixage	Inutile
Protection de l'intégrité des données	CRC-32	Michael (MIC)	CCM
Protection de l'intégrité de l'en-tête	Néant	Michael (MIC)	CCM
Protection contre le rejeu	Néant	Séquence IV	Séquence IV
Gestion des clés	Néant	Basée sur EAP	Basée sur EAP

V. Architecture de sécurité 802.16 et 802.16e (WMAN)

La norme intitulée *Air Interface for Fixed Broadband Wireless Access Systems* [802.16_04] connue sous le nom WiMax est apparue en 2004 pour étendre la connectivité IP sans fil à l'échelle d'un campus. Une année plus tard, des propositions ont été faites à travers le standard émergeant 802.16e intitulé *Air Interface for Fixed and Mobile Broadband Wireless Access Systems* [802.16e_05] pour améliorer le 802.16 dans le sens de la prise en compte des terminaux des abonnés se déplaçant à l'allure d'un véhicule (WiMax mobile).

Au sein de son architecture de sécurité, on retrouve le protocole PKM (*Privacy Key Management*) attaché à la gestion des clés et autour duquel gravitent d'autres protocoles dédiés aux services d'autorisation, d'authentification et de chiffrement (voir figure 3-7).

Le protocole PKM tient compte de l'authentification mutuelle, mais également de l'authentification unilatérale qui correspond au cas où le point d'accès désigné par BS (*Base*

station) est seul à authentifier la station de l'abonné nommée SS (*Subscriber Station*). Il supporte aussi la ré-authentification/ré-autorisation et le rafraîchissement périodiques des clés. Il emploie pour l'authentification soit EAP (possibilité d'emploi des méthodes comme EAP-TLS et EAP-SIM), soit les certificats numériques X.509 en combinaison avec les algorithmes de chiffrement RSA à clé publique, soit encore une combinaison RSA/EAP (séquence débutant avec une authentification RSA et se poursuivant avec une authentification EAP). 3DES, AES et RSA sont les principales méthodes de chiffrement qu'il emploie. L'intégrité des paquets échangés repose sur le calcul de différentes empreintes chiffrées HMAC (*keyed-Hash Message Authentication Code*) et CMAC (*Cipher-based Message Authentication Code*).

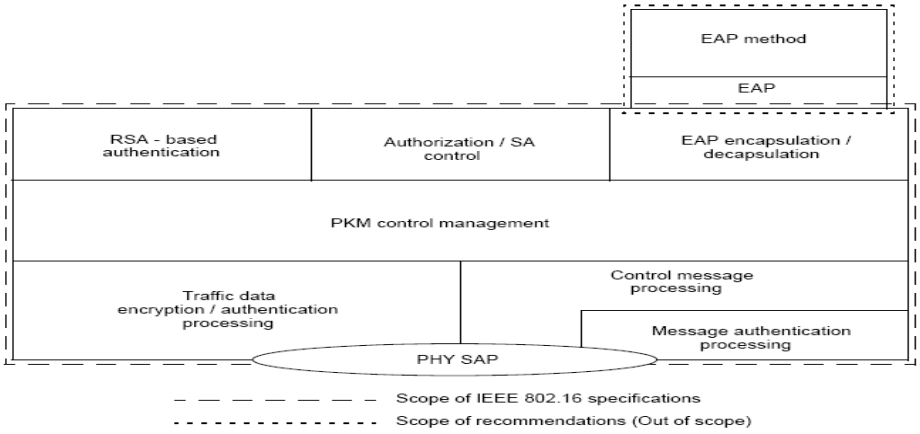


figure 3-7 : éléments de la pile protocolaire de sécurité dans le 802.16e.

En effet, à l'arrivée dans un réseau 802.16e, une SS (client PKM) demande une clé AK (*Authorization Key*) à la BS (serveur PKM) après fourniture de son certificat X.509, de son adresse MAC et de ses capacités (suites cryptographiques). Une fois ces pièces vérifiées et la SS authentifiée par la BS, celle-ci renvoie chiffrée à l'aide de la clé publique de la SS la fameuse clé AK qui constitue le secret partagé dorénavant entre la BS et la SS, et à partir duquel le protocole PKM dérive la plupart des clés qu'il emploie et qui peuvent être classées comme suit :

- des clés de chiffrement utilisées uniquement au moment de l'acheminement des clés de la BS vers la SS authentifiée appelée clé KEK (*Key Encryption Key*). De même, dans le cas de la transmission des clés de la BS vers un groupe particulier de SS, des clés de chiffrement particulières, les GKEK (*Group Key Encryption Key*), sont employées ;
- des clés de chiffrement des données, généralement attribuées par paires, et permettant d'assurer la confidentialité entre la BS et la SS. Elles sont communément désignées sous le nom de TEK (*Traffic Encryption Key*). Une clé TEK est associée à chaque type de flux (association sécurisée) entre la BS et une SS donnée. Elles ont une durée de vie limitée et doivent par conséquent être périodiquement renouvelées. De façon similaire dans une communication multicast, la confidentialité du trafic de la BS vers les SS d'un même groupe est assurée par un chiffrement utilisant une clé GTEK (*Group Traffic Encryption Key*) ;
- des clés spécialement attachées au chiffrement des flux des services multicast et de diffusion MBS (*Multicast and Broadcast Services*) appelées MTK (*MBS Traffic Key*) ;

- des clés destinées à la protection des messages de gestion au moyen de tuples HMAC/CMAC, que ce soit dans le sens descendant de la BS vers une SS (HMAC_Key_D, CMAC_Key_D, HMAC_Key_GD, CMAC_Key_GD,) ou l'inverse (HMAC_Key_U, CMAC_Key_U). Dans le cas particulier de messages EAP échangés au moment de l'authentification, c'est la clé EIK (*EAP Integrity Key*) qui est employée dans le calcul des condensés.

On retrouvera en annexe I le mode de production de ces différents matériaux de clés, un mode qui fait beaucoup usage de la fonction de distribution de clés Dot16KDF.

VI. Récapitulatif des principales caractéristiques des réseaux sans fil IP

Après une demie douzaine d'années d'efforts, les comités de standardisation ont proposé des solutions pour renforcer la sécurité de l'accès aux réseaux sans fil IP, à travers les standards comme WEP, 802.1X, 802.11i et 802.16/802.16e. Les grands traits de ces solutions sont :

- l'absence de module de sécurité spécialement lié à l'utilisateur. La sécurité est construite sur des dispositifs (stations, points d'accès et serveurs d'authentification) dont le contenu (données et codes exécutables/interprétables) peut être corrompu ;
- les paramètres attachés à la mobilité des utilisateurs sont plutôt conservés sur les stations qu'ils emploient ;
- les solutions restent ouvertes, à la différence de ce qui se passe dans les réseaux d'opérateurs. Pour des réseaux sans fil IP, par nature ouverts, les solutions retenues s'appuient sur des méthodes et algorithmes notoirement connus comme résistants aux attaques ;
- les services garantissant la confidentialité et l'intégrité des données échangées sur la radio sont principalement assurés en employant des clés symétriques de moins en moins statiques ;
- une authentification mutuelle robuste est de plus en plus à l'origine de la clé prise comme racine de toutes les clés employées par le demandeur d'accès et le point d'accès pour sécuriser leurs échanges. Ceci permet de bénéficier partout de la sécurité des services de données (Web, messagerie électronique) ou multimédia (voix et images) ;
- un dénominateur commun apparaît entre le 802.1X [802.1X_01] et le 802.16 [802.16_04] en ce sens qu'ils emploient tous le protocole EAP [EAP04] qui est un protocole en mesure d'encapsuler une multitude de méthodes d'authentification.

VII. Conclusion

Dans ce chapitre nous avons présenté les architectures et mécanismes proposés par les principaux standards concernant les réseaux sans fil IP en vue de renforcer la sécurité de leur propre accès mais également celui des services hébergés. Des clés et algorithmes cryptographiques de plus en plus robustes sont proposés et permettent de réaliser de l'authentification mutuelle forte. Malheureusement les machines conservant ces clés sont susceptibles d'être corrompues (attaques du système d'exploitation par des virus, des vers, etc.). De plus, comme l'utilisateur doit être en mesure de retrouver, où qu'il soit et à partir

de n'importe quel terminal de connexion, son profil habituel de connexion et de travail, ces standards n'offrent pas de sécurité pour l'utilisateur des réseaux sans fil IP.

L'introduction et l'utilisation des cartes à puce comme modules de sécurité dans le contexte d'accès aux réseaux sans fil IP nous paraissent constituer des pistes intéressantes de renforcement de la sécurité dans les réseaux sans fil IP, d'autant plus que les premiers résultats des travaux d'implémentation du protocole EAP dans la carte à puce sont prometteurs.

Chapitre 4 : Le protocole EAP et la carte à puce

Nous avons vu lors de la présentation de la sécurité des réseaux sans fil IP à travers les standards 802.1X, 802.11i et 802.16e que le protocole EAP (*Extensible Authentication Protocol*) [EAP04] est retenu pour assurer le transport des méthodes d'authentification. Dans ce chapitre, nous voulons tout d'abord présenter ce protocole et certaines des méthodes d'authentification robustes utilisables dans le contexte des réseaux sans fil. Ensuite, nous décrivons la première implémentation de ce protocole au sein d'une carte à puce Java. Nous terminons le chapitre en présentant les principales contraintes liées à EAP dans une carte à puce.

I. Le protocole EAP

Standardisé en 1994 à l'IETF, le protocole PPP (*Point to Point Protocol*) [RFC1661] permet d'accéder à distance à un réseau local filaire comme Ethernet [IEEE802.3]. Cinq années plus tard, le protocole EAP a été conçu pour permettre aux nombreux clients des fournisseurs d'accès de continuer à se connecter en sécurité à l'Internet à partir de leur ligne téléphonique en utilisant le protocole PPP. EAP a ensuite été introduit dans l'architecture 802.1X définissant un environnement AAA (*Authentication, Authorization, Accounting*) pour les réseaux filaires et sans fil. Le standard émergeant 802.16e, qui peut être considéré comme celui d'un réseau d'accès Internet sans fil mobile, emploie aussi EAP à travers le protocole PKM-EAP (*Privacy Key Management Protocol*) [802.16_04] servant lors de l'authentification des clients de ce réseau. Enfin, depuis qu'il est supporté par le protocole IKEv2 [IKEv2_05], EAP peut être employé pour ouvrir des tunnels sécurisés IPsec [IPSEC98] ou PPTP [PPTP99].

I.1. Contextes de déploiement du protocole EAP

EAP est un protocole qui fonctionne directement sur les couches liaison comme PPP ou IEEE 802, sans faire appel au protocole IP. EAP fournit son propre support pour éliminer les paquets dupliqués ou retransmis, mais est tributaire des garanties offertes par les couches basses. La fragmentation n'est pas supportée à l'intérieur d'EAP lui-même ; cependant les méthodes EAP prises individuellement doivent, si nécessaire, la supporter.

Un dialogue EAP implique généralement deux entités : le client EAP et le serveur EAP. Dans certains contextes, une troisième entité y intervient en tant que relais EAP (figure 4-1).

Le client EAP est généralement hébergé dans le terminal cherchant à se connecter à un réseau donné. Le terminal porte différents noms en fonction des spécifications. Ainsi il est appelé *Suppliant* (demandeur d'accès) dans le standard 802.1X, *Peer* dans le 802.11, *Station* dans le 802.11i, et enfin SS (*Subscriber Station*) dans le 802.16. Le client EAP est chargé d'analyser et de produire les réponses aux requêtes EAP qui lui parviennent.

Le serveur EAP se retrouve sur le dispositif qui, dans une infrastructure réseau, décide de la conduite à tenir par rapport à un client désirant utiliser les services du réseau : le serveur d'authentification. Le serveur EAP est chargé, d'une part de produire les requêtes destinées au client EAP, et d'autre part d'analyser les réponses EAP qui lui sont renvoyées.

Le relais EAP applique la décision d'authentification à l'entrée du réseau auquel veut accéder le client. Dans le standard 802.1X il est appelé *Authenticator*, authentifiant dans notre travail, *Access Point* (AP) dans le 802.11, *Base Station* (BS) dans le 802.16e. Il reçoit du client les pièces à conviction qui permettront à une autorité tierce de décider de la suite à donner à la demande d'accès. La présence de cet élément est optionnelle, et il peut être associé au serveur d'authentification sur la même machine physique (voir figure 4.1).

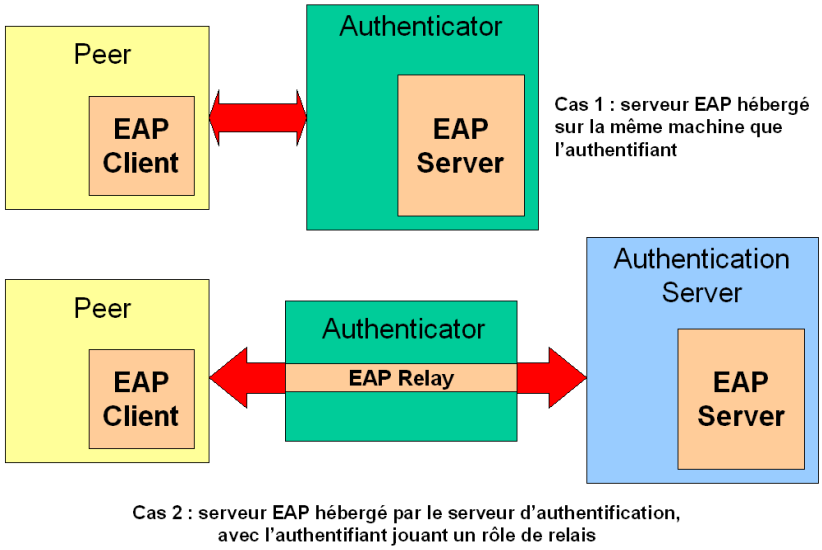


figure 4-1 : contextes de déploiement du protocole EAP.

La flexibilité offerte par le protocole EAP permet la mise à disposition et l'emploi d'un large éventail de mécanismes d'authentification pas forcément négociés à l'avance, indépendants des applications qui les utilisent, et pas obligatoirement déployés au niveau de l'authentifiant. Plutôt que d'exiger de l'authentifiant une mise à jour permanente afin de supporter chaque nouvelle méthode d'authentification, EAP permet l'emploi d'un serveur d'authentification au niveau duquel quelques-unes ou toutes les méthodes d'authentification sont implémentées (figure 4-1, cas 2). Dans ce cas l'authentifiant agit comme simple relais soit pour une partie ou la totalité des méthodes, soit pour une partie ou la totalité des clients demandant l'accès à ce réseau.

I.2. Structure d'un message EAP

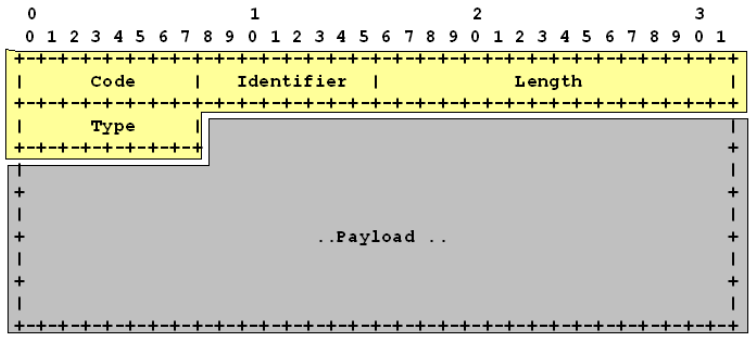
Un message EAP est constitué de deux parties comme l'illustre la figure 4-2 :

- un en-tête de cinq octets utilisés pour les champs *Code* (un octet), *Identifier* (un octet), *Length* (deux octets) et *Type* (un octet) ;
- une charge utile (*Payload*) renfermant les informations en rapport avec un schéma d'authentification particulier.

Le standard suggère que la charge utile soit dans sa structure composée d'un préfixe sur un octet dont l'interprétation permet la bonne récupération des données qui l'accompagnent.

Le champ *Code* caractérise les fonctions du paquet EAP. Il vaut 1 pour *Request*, 2 pour *Response*, 3 pour *Success* et 4 pour *Failure*. Le champ *Identifier* est un numéro qui

permet de mettre en correspondance les requêtes et les réponses. Le champ *Length* donne la taille totale du paquet EAP. Le champ *Type* caractérise la structure d'une requête ou d'une réponse EAP ; toutes les implémentations de EAP supportent les valeurs du champ *Type* allant de 1 à 4 : 1 pour l'identité, 2 pour la notification, 3 pour l'accusé de réception négatif NAK (uniquement pour un paquet de réponse) et 4 pour le challenge MD5. D'autres valeurs du champ *Type* permettent d'identifier le scénario d'authentification.



-Code : 1=Request ; 2=Response ; 3=Success ; 4=Failure
-Type : numéro correspondant au mécanisme d'authentification
-Identifier : identificateur de l'échange
-Length : taille du paquet
-Payload : charge utile du paquet

figure 4-2 : schéma d'un paquet EAP.

I.3. Procédure d'authentification EAP

L'authentification EAP se déroule généralement de la manière suivante :

1. L'authentifiant envoie une requête (paquet *EAP-Request*) demandant au client de s'identifier. Dans cette requête on retrouve une valeur du champ *Type* indiquant ce qui est demandé au client (une identité, un challenge, etc.). Cette demande d'identité initiale n'est pas exigée, et peut être court-circuitée comme dans le cas où l'identité est dépendante du port sur lequel le client se connecte (ligne louée, port de commutateur ou d'appel téléphonique distant, identité obtenue via celle de la machine appelante ou de l'adresse physique MAC, etc.).
2. Le client répond à la requête précédemment reçue en envoyant un paquet *EAP-Response*. Parmi les champs du paquet de réponse on trouve un champ *Type* dont la valeur correspond à celle du champ *Type* de la demande, et assurément la valeur de l'identité réclamée.
3. L'authentifiant vérifie que la réponse reçue correspond à la demande initialement envoyée car EAP fonctionne "pas à pas" ; il ne permet pas d'envoyer une nouvelle requête sans qu'une réponse valide à la précédente requête n'ait été reçue. En cas de réponse valide l'authentifiant l'achemine vers le serveur d'authentification qui la traite. Ce serveur d'authentification peut avoir besoin d'informations complémentaires de la part du client ; dans ce cas il le signifie à l'authentifiant qui, à son tour, transmet la demande correspondante au client. Ainsi, un dialogue entre le client et le serveur d'authentification s'établit, avec l'authentifiant comme relais. Au cas où une requête adressée au client reste sans suite, elle est retransmise au client un certain nombre de fois après lesquelles

l'authentifiant met fin à la conversation EAP mais ne délivre au client aucun message.

4. La conversation d'authentification peut continuer jusqu'à ce que l'authentifiant reçoive de la part du serveur d'authentification une réponse d'authentification. En cas de réponse d'authentification positive (succès), l'authentifiant transmet un paquet *EAP-Success* au client. Dans le cas contraire un paquet *EAP-Failure* lui est transmis.

Le protocole EAP est un protocole d'égal à égal (*peer to peer*). De ce fait une authentification indépendante et simultanée peut se faire dans les deux sens. Chacun des participants à l'échange peut agir en même temps comme celui qui applique et comme celui qui subit l'authentification.

I.4. Modèle de multiplexage EAP

Conceptuellement, une entité EAP est décrite comme constituée d'une pile protocolaire faite de quatre couches comme indiqué à la figure 4-3 :

- La couche basse *Lower layer* responsable de la réception et de la transmission des paquets EAP échangées entre le client et le serveur d'authentification EAP.
- La couche *EAP-Layer* qui s'occupe de la détection des paquets dupliqués, de la retransmission des paquets perdus, de la réception (respectivement envoi) des paquets en provenance (respectivement destination) des couches *EAP-peer Layer* et *EAP-Auth. Layer*. Sur la base de la valeur du champ *Code* s'effectue ici le premier multiplexage, c'est-à-dire l'acheminement vers la couche *EAP-peer Layer* (cas d'une requête reçue) ou la couche *EAP-Auth. Layer*(cas d'une réponse reçue).

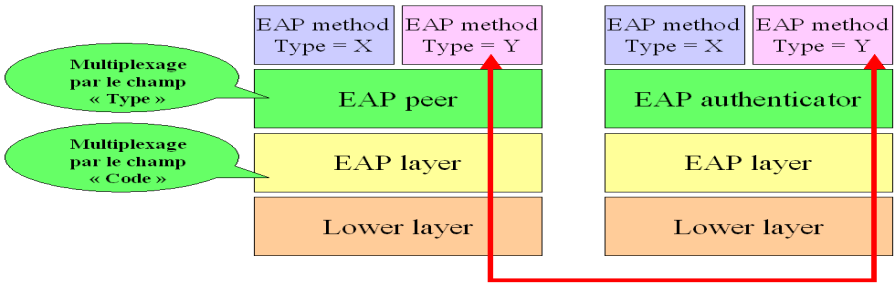


figure 4-3 : modèle de multiplexage EAP.

- Les couches *EAP peer* (côté client) et *EAP-Auth. Layer* (côté serveur) qui abritent le second niveau de multiplexage en ce sens qu'elles acheminent les paquets EAP vers la méthode d'authentification appropriée, en se fondant sur la valeur du champ *Type*.
- La couche *EAP method* hébergeant les méthodes EAP chargées de la réalisation des différents scénarii d'authentification. Chacun d'entre eux est repéré par une valeur du champ *Type* attribuée par l'IANA (*Internet Assignment Numbers Authority*).

I.5. Hiérarchie des clés EAP

Toute authentification mutuelle réussie entre un demandeur d'accès et un serveur d'authentification produit une clé principale dite clé MSK (*Master Session Key*) ainsi qu'une clé dite EMSK (*Extended Master Session Key*) . A partir de la clé MSK est fabriquée la clé

PMK (*Pairwise Master Key*) connue encore sous le nom de clé AAA (*Authentication, Authorization, Accounting*) qui est mise à la disposition de l'authentifiant (figure 4-4) par le serveur d'authentification.

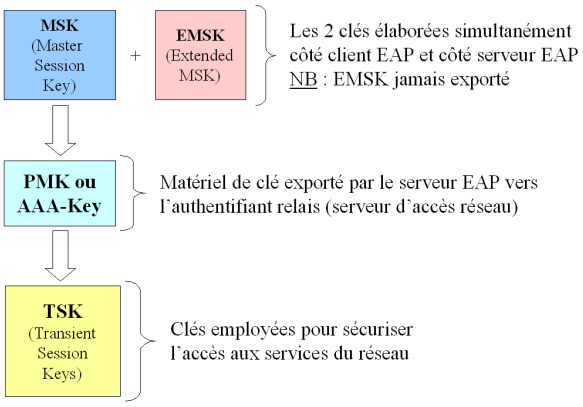


figure 4-4 : hiérarchie des clés dans EAP.

A partir de la clé PMK, maintenant partagée par le demandeur d'accès et l'authentifiant (le point d'accès ou la station de base par exemple), seront dérivées toutes les autres clés servant à sécuriser les échanges entre ces deux entités sur l'interface radio. Ces clés seront généralement appelées TK (*Transient Key*) ou TSK (*Transient Session Key*).

Dans [EAP04] est définie une clé dite EMSK (*Extended Master Session Key*). De taille au moins égale à 64 octets, cette clé produite par la méthode d'authentification n'est jamais partagée avec un tiers. En 2006 son utilisation reste encore à définir, mais est réservée pour le futur.

I.6. Conditions d'emploi du protocole EAP en milieu non sécurisé

Dans les réseaux dont le support physique n'est pas sécurisé, comme dans les réseaux sans fil, un attaquant peut accéder au support physique pour réaliser les attaques suivantes :

- Découverte de l'identité des utilisateurs grâce à l'espionnage des trames échangées.
- Modification ou usurpation de l'identité des paquets EAP.
- Déni de service empêchant la réalisation de nouvelles sessions EAP ou terminant celles en cours.
- Récupération des passe-phrases ou/et mots de passe grâce à une attaque hors-ligne du type du dictionnaire.
- Attraction d'un client d'un réseau sans fil afin qu'il se connecte à un réseau non digne de confiance.
- Perturbation de la négociation EAP afin d'affaiblir l'authentification, d'obtenir les mots de passe des utilisateurs, et de supprimer la protection de la confidentialité.
- Exploitation des faiblesses des techniques de dérivation de clés et/ou des algorithmes employés dans les méthodes EAP.

Afin de se protéger contre ces attaques, le respect des conditions suivantes est obligatoire lors de l'utilisation du protocole EAP :

- Assurer l'authentification mutuelle des deux extrémités de la communication afin de se prémunir des attaques des faux *Authenticators*.
- Assurer aux paquets EAP d'une part une protection de leur intégrité, de leur confidentialité et de leur authenticité, et d'autre part une résistance aux attaques par rejeu.
- Dériver les clés permettant d'assurer l'authentification de chacun des paquets, la protection de leur intégrité et de leur confidentialité, la défense contre le rejeu. De la clé principale de session, des dérivations permettent d'obtenir ultérieurement de nouvelles clés, indépendamment des contextes cryptographiques.
- Résister aux attaques du dictionnaire partout où l'authentification par mot de passe est employée, en évitant l'emploi de mots de passe faibles. Sans cette protection, un attaquant reniflant le trafic d'authentification peut rassembler un grand nombre de messages échangés, et de ce fait obtenir une bonne partie des mots de passe employés. Une attaque est dès lors vite réussie et sans grand frais du fait de la continuelle baisse du coût de la puissance de calcul.
- Donner la possibilité d'une re-connexion à chaud permettant d'écourter de futures phases d'authentification après qu'une conversation antérieure complète d'authentification réussie ait eu lieu.
- Supporter les indications de succès et d'échec partout où les méthodes EAP sont utilisées sur des supports non fiables. Ceci aide à mieux interpréter l'état de la conversation par le client et par l'authentifiant.

II. Des méthodes d'authentification

Plus d'une cinquantaine de méthodes EAP font déjà l'objet d'un enregistrement par l'IANA (*Internet Assignment Numbers Authority*). Nous ne présentons dans ce sous-chapitre que quatre d'entre elles qui ont été par la suite utilisées dans notre travail : EAP-TLS, EAP-SIM, EAP-AKA et EAP-PSK.

II.1. Les protocoles TLS et EAP-TLS

Le protocole TLS (*Transport Layer Security*) [TLS99] est l'un des protocoles de sécurité de la couche transport les plus déployés sur le Web et les plus souvent rencontrés dans le contexte EAP. Cette situation résulte très probablement de la présence de façon native du protocole TLS au sein des serveurs Web. TLS a été défini par l'IETF (*Internet Engineering Task Force*) à partir de la version 3.0 du protocole SSL (*Secure Socket Layer*) [SSLv3_96] principalement employé pour la sécurité des applications du Web. Placé habituellement entre la couche TCP [STEV94] et les protocoles de la couche application, il permet selon un modèle de fonctionnement client /serveur de :

- Toujours authentifier le serveur vis-à-vis du client, et optionnellement d'assurer l'authentification du client vis-à-vis du serveur, sur la base d'une suite cryptographique négociée.
- Mettre en place une connexion confidentielle et intègre, fondée sur une clé de session produite à la suite de l'authentification.

On trouvera davantage de précisions sur SSL/TLS dans le livre de Rescorla [RESCO03]. Si dans le contexte général, seule l'authentification du serveur par le client est exigée, nous voulons considérer dans notre approche sécurisée (authentification mutuelle) que systématique est également l'authentification du client par le serveur. Ainsi, une session TLS réussie dans notre contexte sera toujours caractérisée par au moins :

- un identificateur de session généré par le serveur ;

- une suite cryptographique retenue ;
- une clé cryptographique MSK générée.

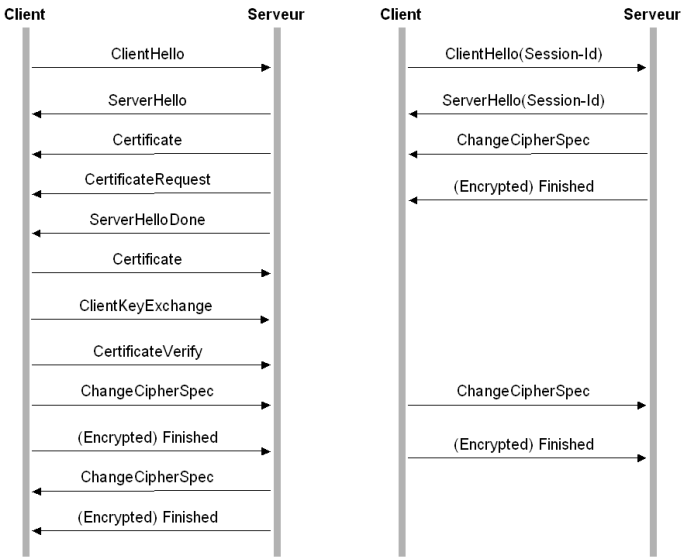


figure 4-5 : authentification TLS en "*full mode*" à gauche, et en "*resume mode*" à droite.

Dans notre travail, une session TLS se décompose en deux phases : celle de l'ouverture de la session par une authentification mutuelle, suivie de celle de l'échange des données. Comme présentées à la figure 4-5 deux procédures d'authentification par TLS existent : la procédure complète dénommée *Full mode TLS*, et la procédure de reprise dite *Resume mode TLS*.

Dans une procédure d'authentification complète, la phase d'ouverture de la session TLS est démarrée par le client à l'origine d'un message *ClientHello* contenant principalement un nombre aléatoire R_1 et la liste de suites cryptographiques supportées. Ces suites cryptographiques se rapportent aux algorithmes d'échange de clés (RSA, DH), aux algorithmes de chiffrement (RC4, DES, 3DES, etc.), aux algorithmes de hachage (MD5, SHA-1), aux algorithmes de signature RSA [RSA78] et DSA[NIST97]).

Le serveur TLS à son tour, répond par un premier message *ServerHello* renfermant entre autres choses un nombre aléatoire R_2, un identificateur de session (soit ancien, soit nouveau), la suite cryptographique retenue. Un deuxième message *Certificate* permet au serveur de prouver son identité en envoyant son certificat numérique X.509 [PKICRL99] qui peut optionnellement inclure la chaîne complète de certificats remontant jusqu'à l'autorité de certification (CA) racine. Les certificats sont vérifiés en contrôlant les dates de validité et en s'assurant qu'ils portent la signature d'une autorité de certification approuvée. Un troisième message *CertificateRequest* permet de demander au client son certificat X.509, après quoi le serveur par un dernier message *ServerHelloDone* redonne le contrôle au client.

Le client TLS alors répond en renvoyant tout d'abord son certificat par un premier message *Certificate*. Ensuite par l'intermédiaire du message *ClientKeyExchange* il transmet chiffrée au serveur sa contribution PMK (*Pre-Master secret Key*) à la fabrication de la clé maîtresse MSK (*Master Secret Key*) et de toutes les autres clés que chacune des deux entités aura à dériver. Afin de permettre au serveur de l'authentifier, le client envoie par le biais

d'un message *CertificateVerify* une signature du condensé de tous les messages échangés jusqu'alors. Un avant dernier message envoyé par le client et dénommé *ChangeCipherSuite* indique au serveur l'emploi dorénavant du contexte cryptographique négocié. Par un dernier message appelé *Finished* constitué des condensés MD5 (HMAC-MD5) et SHA1 (HMAC-SHA1) de tous les messages échangés, le client rend la main au serveur.

Par l'envoi d'un message *ChangeCipherSuite*, le serveur à son tour indique au client qu'il va utiliser à partir de cet instant le contexte cryptographique négocié. L'envoi par le serveur d'un message *Finished* renfermant les condensés MD5 et SHA1 de tous les messages échangés met fin à cette phase d'authentification. Si de part et d'autre les vérifications sont satisfaisantes, l'échange de données au travers d'un canal sécurisé peut alors s'effectuer.

Quant à la procédure d'authentification abrégée *TLS mode resume* (partie droite de la figure 4-5), elle correspond à la reprise d'une session complète antérieurement réalisée et dont le couple (client, serveur) a conservé les paramètres d'identification (identificateur de session et clé MSK). Elle évite plusieurs opérations cryptographiques telles que le décodage et la vérification des certificats du client/serveur, la consultation des listes de révocation de certificats, la génération et le chiffrement/déchiffrement de l'avant première clé secrète PMK. Elle réduit aussi le flux des messages.

En définitive, on peut affirmer que dans notre contexte le protocole TLS fournit une sécurité de connexion possédant les trois propriétés suivantes :

- une authentification mutuelle des entités communicantes ;
- un canal d'échanges confidentielles de données dès la fin de l'authentification ;
- une connexion fiable fondée sur un transport de messages incluant un contrôle d'intégrité par HMAC.

Par conséquent, EAP-TLS [EAPTLS99] peut se définir comme le protocole d'authentification mutuelle TLS réalisé au-dessus de EAP et qui place chaque message TLS dans un paquet EAP-TLS. Lorsque le dialogue d'authentification TLS réussit, l'authentifiant en est informé et reçoit de la part du serveur d'authentification la clé de session AAA à partir de laquelle vont dériver toutes les clés servant à la sécurité des échanges entre le demandeur d'accès et lui.

EAP-TLS est la méthode d'authentification de facto de l'infrastructure 802.11i étant donné qu'elle permet de satisfaire toutes les exigences de cette infrastructure, ce qu'aucune autre méthode aussi répandue n'est en mesure de faire.

II.2. Le protocole EAP-SIM

La méthode d'authentification EAP-SIM est spécifique aux cartes SIM utilisées dans les réseaux GSM. Elle sert à renforcer les mécanismes originels d'authentification et de génération/distribution de clés de session du GSM par une combinaison de n (n=1, 2 ou 3) triplets (RAND, SRES, Kc) supposés mis à la disposition du serveur d'authentification EAP.

Lors d'une authentification complète par la méthode EAP-SIM, une clé racine MK (*Master Key*) est dérivée à partir des clés GSM Kc, des valeurs aléatoires et d'autres données contextuelles comme suit :

MK = SHA1 (*Identity* | *n*Kc* | *NONCE_MT* | *Version_List* | *Selected_Version*),

avec *Identity* correspondant à la chaîne de caractères associée à l'identité de l'utilisateur qui peut être soit l'IMSI (*International Mobile Subscriber Identity*), soit un NAI (*Network Access Identity*) [NAI05] ou un pseudonyme, *n*Kc* associé aux n valeurs de Kc concaténées dans l'ordre, avec chaque clé Kc résultant de l'application de la fonction GSM A8 aux deux

entrées que sont le nombre aléatoire envoyé par le réseau au demandeur d'accès et la clé Ki, *NONCE_MT* correspondant à un nombre aléatoire généré par le client, *Version_List* désignant la liste des numéros de versions EAP-SIM supportées par le serveur d'authentification, *Selected_Version* correspondant à la version EAP-SIM retenue.

La clé MK est ensuite fournie à un générateur de nombres pseudo-aléatoires (PRF) qui produit :

- deux clés TEK (*Transient EAP Keys*) destinées à la protection des paquets EAP-SIM et qui sont la clé d'authentification K_aut et la clé de chiffrement K_encr ;
- une clé MSK (*Master Session Key*) destinée à la protection du trafic entre le client et l'authentifiant,
- un vecteur d'initialisation IV (*Initialization Vector*).

Lors d'une ré-authentification les mêmes clés TEK vont être employées pour protéger les paquets EAP, mais une nouvelle clé MSK et de nouveaux vecteurs d'initialisation IV seront dérivés de la clé MK initiale combinée avec de nouvelles valeurs aléatoires échangées durant cette ré-authentification.

Notons enfin que la clé MSK de 128 octets et les deux vecteurs d'initialisation IV de 32 octets dérivés du protocole EAP-SIM peuvent également servir de matériels de clés pour EAP-TLS. L'obtention de clés compatibles EAP-TLS découle du découpage de tout ce matériel de clés en six blocs de 32 octets employés comme suit : le 1er bloc pour la clé de chiffrement du client, le 2e bloc comme clé de chiffrement du serveur d'authentification EAP, le 3e bloc comme clé d'authentification du client (employé dans le calcul des MAC), le 4e bloc comme clé d'authentification du serveur d'authentification, le 5e bloc comme vecteur d'initialisation du client, et le 6e bloc comme vecteur d'initialisation du serveur.

II.3. Le protocole EAP-AKA

Le protocole EAP-AKA [EAPAKA06] est une méthode d'authentification et de distribution de clés de session définie pour l'UMTS (*Universal Mobile Telecommunications System*) et le CDMA2000, à la manière de ce qu'est le protocole EAP-SIM pour le GSM.

Il est une adaptation de l'algorithme symétrique Milenage [MILEN02] où une session d'authentification complète n'est constituée que d'une requête et d'une réponse. La figure 4-6 récapitule assez bien cet échange en faisant apparaître les différents calculs sous-jacents.

Le protocole EAP-AKA suppose que le module d'identité (carte USIM pour l'UMTS) et le serveur d'authentification partagent un secret ; son fonctionnement peut être décrit de la manière suivante :

1. L'authentification démarre avec la production, du côté du serveur d'authentification, d'un vecteur d'authentification calculé à partir du secret partagé et d'un numéro de séquence. Ce vecteur d'authentification renferme un nombre aléatoire RAND, une valeur d'authentification AUTN permettant d'authentifier le serveur d'authentification lui-même auprès de la carte USIM, un résultat de chiffrement escompté XRES, une clé IK (*Integrity check Key*) de 128 bits destinée au calcul du MIC (*Message Integrity Code*) et enfin une clé CK (*Cipher Key*) de 128 bits à employer pour le chiffrement des données échangées durant la session.
2. Les valeurs RAND et AUTN sont transmises au module USIM.
3. Le module USIM vérifie la valeur AUTN reçue en utilisant également le secret partagé et le numéro de séquence. Si la comparaison est concluante, c'est-à-dire que la valeur AUTN est valide et que le numéro de séquence employé pour cela figure

dans la bonne plage, alors le module d'identité produit un résultat d'authentification RES qu'il envoie au serveur d'authentification.

4. Le serveur d'authentification en recevant RES le compare au nombre XRES initialement calculé. S'il y a égalité alors le serveur d'authentification conclut que les clés IK et CK peuvent être employées pour dériver les autres clés.

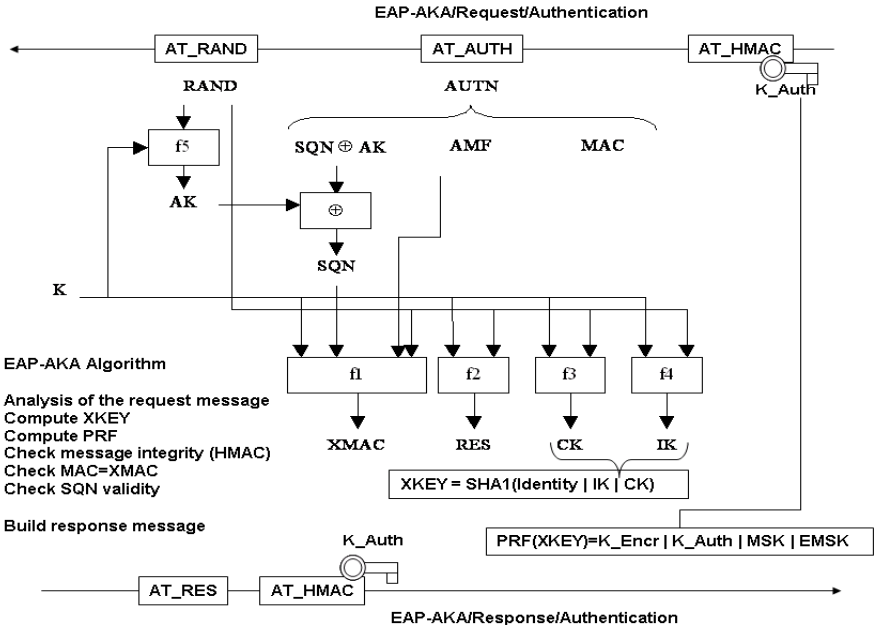


figure 4-6 : résumé d'une authentification normale avec le protocole EAP-AKA.

A l'issue d'une authentification complète EAP-AKA, une clé principale MK (*Master Key*) est dérivée à partir des clés CK, IK, et de l'identité associée à la carte USIM en calculant MK=SHA1(Identity | IK | CK). Et tout comme dans le protocole EAP-SIM, la clé MK introduite dans une fonction PRF servira à générer des clés TEK, MSK et EMSK.

Lors d'une ré-authentification les mêmes clés TEK vont être employées pour protéger les paquets EAP. En revanche, de nouvelles clés MSK et EMSK devront être dérivées de la clé MK initiale en combinaison avec les valeurs aléatoires échangées au cours de cette ré-authentification.

II.4. Le protocole EAP-PSK

Le protocole EAP-PSK (EAP - *Pre-Shared Key*) [EAPPSK04] correspond à un scénario d'authentification symétrique proposé en 2004 à l'IETF et basé sur l'algorithme de chiffrement par bloc AES-128 (*Advanced Encryption Standard*) [NIST01].

Ce protocole est réalisé en trois étapes : la première est celle de la mise en place des clés AK (*Authentication Key*) et KDK (*Key Derivation Key*) à partir du premier matériau cryptographique qu'est la clé PSK supposée uniquement partagée entre le serveur EAP et le client EAP, la deuxième est celle de l'authentification mutuelle entre le serveur et le client EAP pendant laquelle sont dérivées les clés de session TEK (*Transient Encryption Key*), MSK (*Master Session Key*) et EMSK (*Extended MSK*), et enfin la troisième qui est celle de l'échange entre les deux parties à travers un canal sécurisé. Les deux premières étapes emploient comme algorithme le *Modified Counter Mode* [MCM03] basé sur des chiffrements par AES et des opérations de OU exclusif, tandis que l'algorithme EAX employant trois

instances de la fonction OMAC (*One-Key CBC MAC*) [OMAC03] est retenu pour la réalisation de la troisième étape de chiffrement avec garantie de l'intégrité des messages échangés. L'enchaînement des traitements et des matériaux employés lors de la réalisation du protocole EAP-PSK est illustré à la figure 4-7.

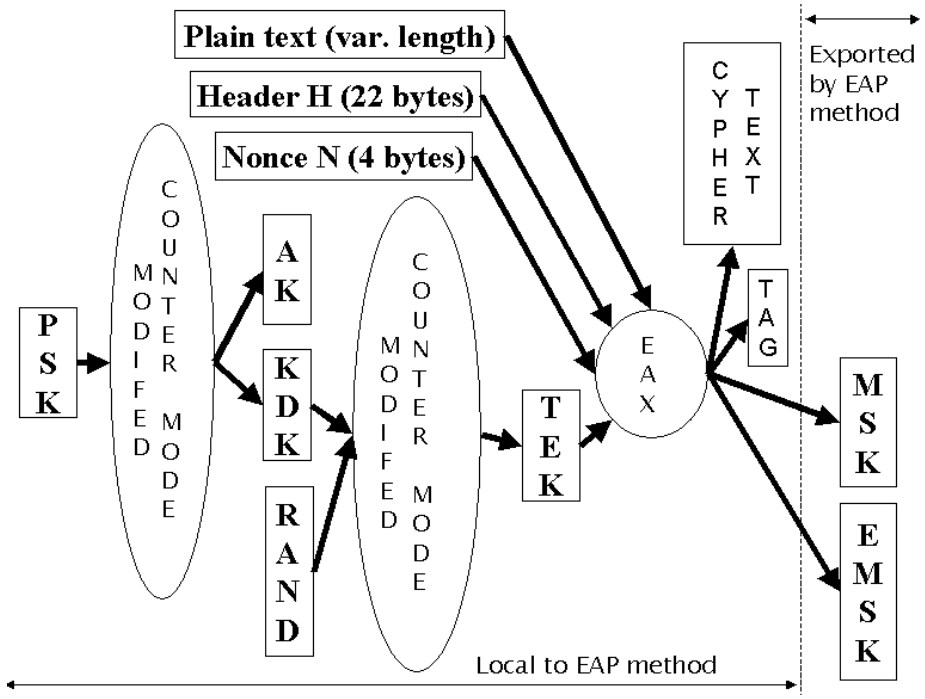


figure 4-7 : schéma de réalisation fonctionnelle de EAP-PSK.

III. Le protocole EAP dans une carte à puce

III.1. Description générale

En 2003 est réalisée la première carte à puce abritant un client EAP [UrLo03]. Dénommée carte à puce EAP, les détails de son implémentation sont fournis dans [EAPSup06]. Assimilable à un micro contrôleur ISO 7816 ouvert, il est capable de supporter la plupart des protocoles d'authentification.

Au lieu que la carte à puce soit simplement vue comme un dispositif d'exécution où les fonctions cryptographiques qu'elle héberge sont invoquées selon les paradigmes traditionnels d'appel de procédure à distance RPC, c'est plutôt comme une actrice dans le déroulement d'un protocole qu'elle a été retenue. Dans cette approche le code du protocole EAP est scindé en deux parties :

- Une première partie dénommée *moteur EAP* se trouve logée dans la carte à puce. Elle a principalement en charge l'analyse des requêtes qui lui arrivent et la production des réponses correspondantes.
- Une deuxième composante plus légère dénommée *entité logicielle EAP* (figure 4-8), est conservée sur le terminal. Elle diffuse les messages EAP échangés entre le réseau et la carte à puce et s'occupe également de détecter et gérer une éventuelle absence de la carte. Par exemple lorsqu'une carte à puce n'est pas installée dans le terminal,

un accusé de réception négatif est renvoyé pour chaque requête reçue par le client EAP.

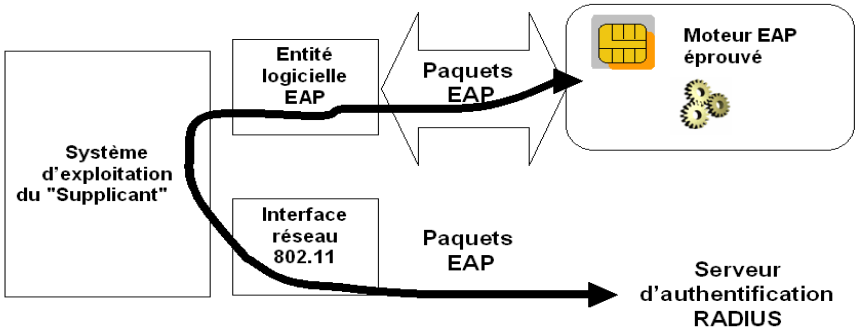


figure 4-8 : logiciel client EAP réparti sur la carte à puce et le terminal.

Le gros avantage de cette approche protocolaire est la création d'un canal de communication entre l'entité que renferme la carte à puce et le serveur d'authentification RADIUS (voir figure 4-9), canal pouvant être employé pour échanger des informations comme :

- Le profil de l'abonné qui correspond à un ensemble de données y compris des certificats. Il présente au réseau sans fil les lettres de crédits (*credentials*) pour les services offerts, ainsi que les préférences du visiteur. Cette collection de données peut être sous différentes formes, par exemple ASN.1 ou XML. Habituellement, ce profil est conservé dans un entrepôt LDAP sollicité à chaque procédure d'authentification. Les cartes à puce EAP pourraient stocker ce profil (par exemple sous la forme d'un certificat X.509), afin de supporter une architecture distribuée qui faciliterait le déplacement (*roaming*) entre différents fournisseurs de services Internet sans fil, en particulier ceux employant l'infrastructure PKI ou ceux n'exigeant pas une connexion à une base de données centralisées.

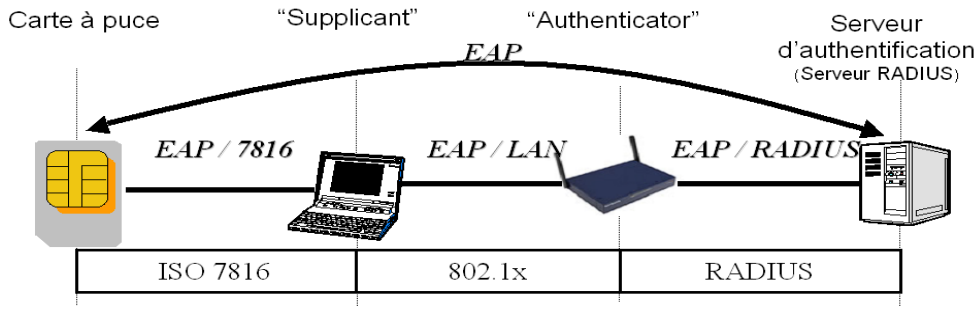


figure 4-9 : contexte protocolaire de la carte à puce EAP.

- Le profil du réseau visité dans lequel peuvent se retrouver des noms de serveurs ou des clés cryptographiques. Par exemple, des protocoles tels que RSVP (*Resource ReSerVation Protocol*) [RSVP97] et COPS (*Common Open Policy Service*) [COPS00] font usage de secrets partagés pour calculer les condensés chiffrés de messages (classiquement une valeur de condensé obtenue à partir du contenu d'un message et d'un mot de passe). Un protocole d'authentification spéciale intégrant la

confidentialité, l'intégrité et la signature numérique pourrait transporter des clés nécessaires à certains services locaux comme la QoS.

- Des données de gestion de la carte à puce. Elles vont par exemple autoriser la modification du profil de l'abonné, ou encore le téléchargement dans la carte à puce d'une nouvelle application telle qu'un fichier de classe Java. Un tel processus est comparable à ce qui se fait habituellement à distance dans le réseau GSM, en utilisant des échanges de messages courts SMS.

III.2. Services de la carte EAP

La carte à puce EAP offre quatre classes de services (figure 4-10) disponibles aux interfaces définies dans [EAPTLS04] et détaillées dans [EAPSup06] ; ce sont :

- La classe des services réseau. On y offre des services de traitement des messages EAP transportés dans des APDUs ISO 7816. En cas de besoin, des clés de session peuvent être calculées et délivrées au système d'exploitation du terminal, des clés utilisées par des protocoles dans l'objectif de sécuriser les échanges sur la radio.
- La classe des services système d'exploitation. Ils rendent possibles la création et la suppression de plusieurs triplets (EAP-ID, EAP-Type, clés cryptographiques) stockés sur la carte à puce, et identifiés par un paramètre appelé *Identité*. Chaque identité peut être associée à un profil pouvant par exemple conserver une liste de SSIDs (*Service Set Identifier*) ou un certificat X509.
- La classe des services de protection des domaines de l'émetteur et du propriétaire de la carte. Deux codes PIN sont disponibles sur la carte ; l'un servant à authentifier le porteur de la carte, et l'autre destiné à l'authentification de l'émetteur de la carte. Il est ainsi donné au propriétaire et à l'émetteur de la carte de protéger les données qu'ils estiment sensibles, par la demande de vérification de ces codes lors de certaines opérations.
- La classe des services de personnalisation des identités. Ici se retrouvent les possibilités de définition et de mise à jour des identités présentes au sein de la carte.

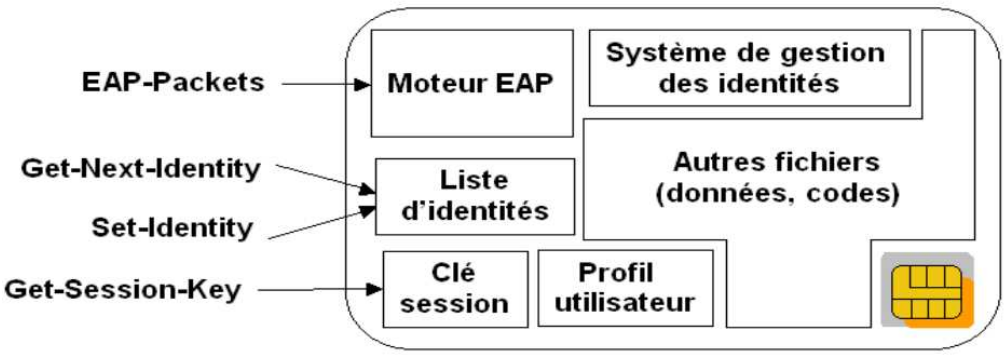


figure 4-10 : services de base de la carte à puce EAP.

L'identité est un concept fondamental au niveau de la carte à puce EAP puisque le processus d'authentification démarre à la réception d'un message de requête d'identité par le demandeur d'accès qui, ici, correspond à la carte à puce EAP. En guise de réponse, ce demandeur d'accès délivre un message de réponse contenant la valeur de l'identité du

système appelée EAP-ID qui va orienter aussi bien vers le scénario d'authentification que vers le contexte de sa réalisation. Elle peut ressembler à :

- Un élément SSID (*Service Set Identifier*) qui, au niveau des réseaux 802.11, est une chaîne de 32 octets identifiant le réseau [802.11_99]. Ce paramètre est obtenu par le système d'exploitation et peut servir comme sélecteur d'identité à mettre en place dans la carte à puce EAP.
- Un identificateur d'utilisateur (*UserID*) pouvant être interprété comme un identificateur d'accès au réseau [NAI05].
- Un pseudonyme (*LocalID*) qui pourrait être par exemple un compte utilisateur dans un environnement particulier de traitement.

En imaginant la même carte à puce EAP employée dans plusieurs classes de réseaux sans fil gérés par des autorités administratives différentes, c'est plusieurs systèmes d'identités qu'il faudra faire coexister dans cet équipement infalsifiable.

III.3. Contraintes à lever par la carte à puce EAP

Deux facteurs majeurs limitent l'emploi des cartes à puce réalisant le protocole EAP et les méthodes qu'il héberge : la complexité du protocole et la vitesse de traitement de la carte.

La complexité du protocole implémenté dans la carte à puce Java doit être compatible avec les ressources de calcul et de stockage qui sont disponibles sur la carte. En termes d'espace mémoire disponible, le *bytecode* associé au protocole EAP et aux méthodes d'authentification supportées doit être en mesure de tenir dans la mémoire non volatile de la carte, mémoire qui généralement est très limitée par rapport à celle des ordinateurs portables classiques.

S'agissant des traitements, trois principales catégories sont à prendre en compte : celle des traitements liés aux transferts de données, celle des opérations cryptographiques, et tous les autres traitements logiciels. En effet, du fait de l'insuffisance de la mémoire RAM disponible, les informations envoyées à la carte à puce sont écrites et lues dans la mémoire non volatile (EEPROM ou mémoire flash). Par conséquent, le temps de transfert des données peut être considéré comme le temps nécessaire au transport des paquets EAP entre le terminal qui contrôle la carte à puce et l'application qui s'exécute dans la carte à puce. Les contributions comptabilisées dans ce temps de transfert sont par exemple le délai de transfert entre le terminal et le lecteur de carte, la durée du transfert entre le lecteur de carte et la carte elle-même, les délais logiciels internes (introduits par exemple par les opérations Java), le temps mis pour les accès à la mémoire.

Dans le contexte de la carte Java, les fonctions cryptographiques sont invoquées à travers des API spécifiques. Les fonctions/procédures cryptographiques comme MD5, SHA1, RSA, AES et PRF, si elles existent de façon native dans le système d'exploitation de la carte à puce, permettent de réaliser assez rapidement les méthodes d'authentification qui les utilisent.

En revanche, tous les autres traitements employés au sein de la carte à puce et non réalisés par des API Java devront l'être grâce à du code Java additionnel. Par exemple l'analyse de certificats X.509 ou le calcul d'un condensé particulier seront généralement obtenus par l'emploi d'un code additionnel interprété par la machine Java présente dans la carte à puce.

La réalisation de tous ces traitements doit respecter des contraintes.

La première est en rapport avec le temps s'écoulant entre l'envoi de la requête du serveur d'authentification, et la réception de la réponse correspondante. En effet, du côté de l'authentifiant, le serveur EAP envoie des requêtes et attend en retour des réponses avant l'échéance d'un délai de 30 s par défaut contrôlée par un temporisateur appelé *txPeriod* [802.1X_01]. Comme ce temporisateur est réarmé trois fois de suite en cas de problèmes, le fait que le temps de traitement d'une requête par la carte à puce dépasse cette valeur d'attente va forcément être à l'origine d'une retransmission.

La seconde contrainte est liée au fait que sur les plate-formes Windows, EAP est lancé en parallèle avec le client DHCP (*Dynamic Host Configuration Protocol*) [DHCP97] présent sur le terminal, et qui permet d'obtenir dynamiquement une adresse IP dès la mise en marche de ce dernier sur un réseau local. Cet événement est déclenché dès lors que l'interface réseau est activée, et cela indépendamment de l'authentification par EAP. Si le client IP ne reçoit pas d'acquittement DHCP dans un délai raisonnable habituel de 60 s, le système d'exploitation du terminal ré-initialise l'interface réseau, puis redémarre les deux processus DHCP et EAP. Il faut donc arriver à traiter un scénario d'authentification en moins de 60 s, un temps pouvant inclure la fourniture du code PIN de l'utilisateur.

IV. Conclusion

Dans ce chapitre nous avons présenté le protocole EAP et les méthodes EAP-TLS, EAP-SIM et EAP-AKA qui toutes sont susceptibles d'être implémentées sur des cartes à puce. Nous avons terminé le chapitre en décrivant la première implémentation du protocole EAP dans une carte à puce Java, avec une indication des contraintes qu'il faut nécessairement surmonter si ce module doit servir à l'authentification.

Les quatre principales contributions de cette thèse sont des réponses par rapport aux limitations de la carte à puce EAP : le protocole EAP-SSC, la plate-forme OpenEapSmartcard, le serveur d'authentification EAP, et enfin l'architecture TEAPM. Chacune de ces contributions fait l'objet d'un des chapitres qui suivent.

CONTRIBUTIONS

Chapitre 5 : Le protocole EAP-SSC

Une fois le protocole EAP implémenté avec succès[6] au sein d'une carte à puce, le problème à résoudre a consisté à rechercher parmi les mécanismes d'authentification disponibles, lequel tout en étant peu complexe, pouvait permettre de réaliser une authentification mutuelle forte, aussi bien dans un contexte de chiffrement symétrique qu'asymétrique. N'ayant à l'époque rien trouvé qui réponde à ces attentes, nous avons proposé un draft à l'IETF intitulé EAP-SSC *(EAP Secured Smartcard Channel)* [draftSSC03] et publié à ce sujet un article [EAPSSC04].

L'objet de ce chapitre organisé en trois parties est de présenter ce nouveau protocole. Dans une première partie nous traitons du cahier des charges auquel doit répondre le nouveau protocole. Ensuite, nous présentons la solution à laquelle nous sommes arrivés. Dans un troisième temps une analyse critique de notre protocole est faite avec des propositions d'amélioration.

I. Cahier des charges du protocole

S'il est vrai que SSL/TLS est l'un des protocoles universellement employés dans le domaine de la sécurisation de la couche transport, il est également notoire que ce protocole est complexe et exige d'importantes ressources de traitement et de mémorisation, ce dont ne dispose encore pas une carte à puce. L'idée est donc de mettre au point un protocole tirant profit des forces de SSL/TLS pour permettre de sécuriser un canal de transport au moyen de la carte à puce.

Les principales caractéristiques que le protocole devrait avoir sont :

1. Se fonder sur le socle EAP de sorte à faire partie de la cinquantaine de mécanismes d'authentification déjà dénombrés sous le "parapluie" EAP ;
2. Assurer une authentification mutuelle avec ou sans partage de secret entre la carte à puce et le serveur d'authentification. En fait, on attend de ce protocole qu'il puisse, aussi bien dans un contexte de partage de secret que dans un contexte de clés publiques, assurer une authentification mutuelle entre deux entités.
3. Offrir à la demande la garantie de confidentialité des messages en utilisant le chiffrement. Cette caractéristique doit rendre possible le transport en clair ou de façon cryptée de la charge utile des messages de ce protocole.
4. Garantir l'intégrité des messages. Ce protocole doit permettre selon le contexte, de mettre en place le service assurant l'intégrité des messages échangés.
5. Assurer la protection contre le rejeu. Ce service est assuré obligatoirement afin de contrer toute réutilisation frauduleuse d'anciennes clés de session.
6. Parer les attaques du type de "l'homme du milieu". C'est également un service obligatoire qui va de pair avec les garanties de confidentialité et d'intégrité.
7. Etre simple mais robuste en optimisant le flux de messages nécessaires à la mise en place de sessions sécurisées.
8. Pouvoir se déployer dans le contexte actuel sans nécessiter des modifications des protocoles qui existent.
9. Nécessiter un code pouvant tenir dans une carte puce.

[6] Technologie innovante récompensée au concours «Cartes 2003» par le prix de la meilleure innovation technologique. http://www.cartes.com/en/frameset_dyn.htm?URL=I_trophee/I4_gagnants.htm.

10. Nécessiter un temps optimum d'exécution d'une session d'authentification de sorte à garantir un temps inférieur à la limite des 60 secondes.

II. Présentation générale du protocole EAP-SSC

II.1. Vue d'ensemble

En plus du draft IETF, le protocole EAP-SSC a été l'objet de deux publications [EAPSSC04] et [CARI04]. Il a été conçu en vue de mettre en place un canal sécurisé entre la carte à puce et un serveur d'authentification.

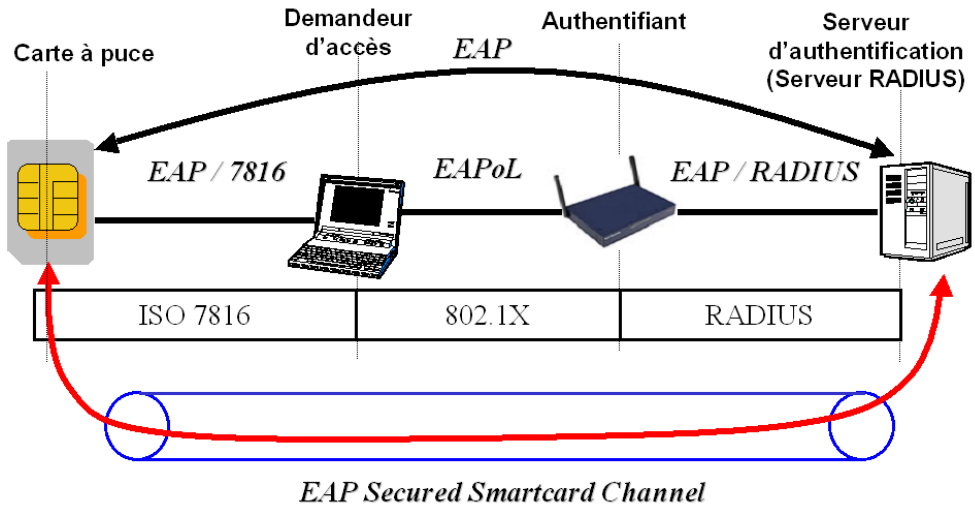


figure 5-1 : environnement protocolaire de la méthode d'authentification EAP-SSC.

Ce canal représenté à la figure 5-1, est créé en s'appuyant sur le protocole EAP présents dans la carte à puce, le terminal hôte de cette carte à puce, le point d'accès jouant le rôle d'authentifiant, et le serveur d'authentification. Comme protocoles de transport de ce protocole EAP, on suppose disposer des protocoles :

- ISO/IEC 7816-4 entre la carte à puce et son terminal hôte,
- EAPoL entre le terminal et le point d'accès de l'architecture sécurisé 802.1X, et
- RADIUS entre le point d'accès et le serveur d'authentification.

II.2. Format du paquet EAP-SSC

A la figure 5-2 est illustré un paquet EAP-SSC encapsulé dans un autre paquet EAP classique. Les champs *Code*, *Identifier*, *Length* et *Type* sont décrits selon [EAP04]. L'emploi de EAP-SSC est subordonné à l'attribution par l'IANA d'une nouvelle valeur pour le champ *Type* qui permettra de distinguer ce nouveau protocole de ceux qui existent déjà.

Le champ *Sub-Type* est codé sur le premier octet du message EAP-SSC. Il permet de faire la distinction entre plusieurs familles de messages ; actuellement *Sub-Type*=1 correspond au choix du contexte de chiffrement symétrique, et *Sub-Type*=2 au modèle cryptographique asymétrique. La présence de ce champ constitue une ouverture par

rapport au futur en ce sens que d'autres familles de messages pourraient être prises en compte sous le même type EAP-SSC.

Le champ *Flags* est codé sur le deuxième octet du message EAP-SSC et complète le champ *Sub-Type*. Ainsi, pour les valeurs 1 ou 2 du *Sub-Type*, les bits du *Flags* ont la signification donnée en bas de la figure 5-2, avec le bit L correspondant au bit de plus fort poids et le bit R comme celui de plus faible poids. Les bits L (*Length included*), M (*More fragments*) et S (*Start*) s'interprètent comme au sein du protocole EAP-TLS. Le bit E (*End*) mis à 1 dans un paquet émis par le serveur d'authentification marque la différence avec les autres messages. Le bit D (*Digest*) mis à 1 dans un paquet signifie sa terminaison par un condensé. Quand le corps du message est chiffré le bit C (*Ciphered payload*) est à 1, et quand il renferme une séquence de certificats X.509 le bit X (séquence de certificats X.509) est à 1. Le bit R signifie "Réservé" et reste à 0 pour le moment.

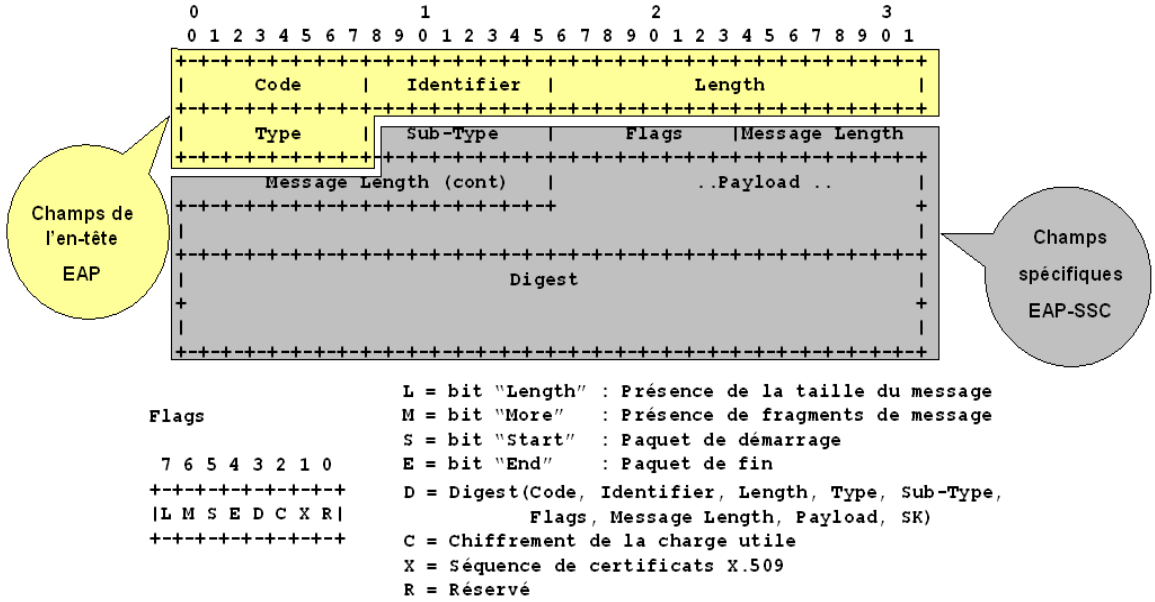


figure 5-2 : format des messages EAP-SSC.

Le champ *Message Length* occupe trois octets figurant dans le message EAP-SSC lorsque la valeur du bit L vaut 1. Il fournit la taille totale du message EAP-SSC ou de l'ensemble des fragments de message auxquels il a donné lieu.

Le champ *Payload* est censé abriter la charge utile du message dépendant des champs *Sub-Type* et *Flags*.

Le *Digest* termine un messsage EAP-SSC quand le bit D du champ *Flags* vaut 1 ; il a 160 bits et correspond au résultat calculé, par défaut selon l'algorithme SHA-1 (*Secure Hash Algorithm*) [SHA1_02], sur une entrée issue de la concaténation d'une liste précise de champs du message.

II.3. Authentification mutuelle par EAP-SSC

Une authentification mutuelle entre la carte à puce et le serveur d'authentification a lieu dès que le serveur d'authentification reçoit de l'authentifiant le paquet *EAP-Response/Identity* dans lequel est encapsulé un enregistrement du type EAP-SSC. Son déroulement est fonction du contexte cryptographique et se décompose en deux étapes :

d'abord une phase de partage et validation d'une clé de session, et ensuite celle d'un échange sécurisé de messages.

La phase de partage et validation de la clé de session est réalisée grâce à deux échanges/transactions dont le serveur d'authentification est toujours l'initiateur. Le premier échange sert au partage de la clé de session, tandis que le deuxième est destiné à sa validation.

Une illustration de ces phases dans les deux contextes cryptographiques est fournie à travers la figure 5-3 et la figure 5-4. Pour des raisons de présentation, nous décidons d'appeler EAPOR (*EAP over RADIUS*) le protocole d'encapsulation de EAP dans RADIUS, un peu à la manière du protocole EAPOL introduit par l'architecture de sécurité 802.1X.

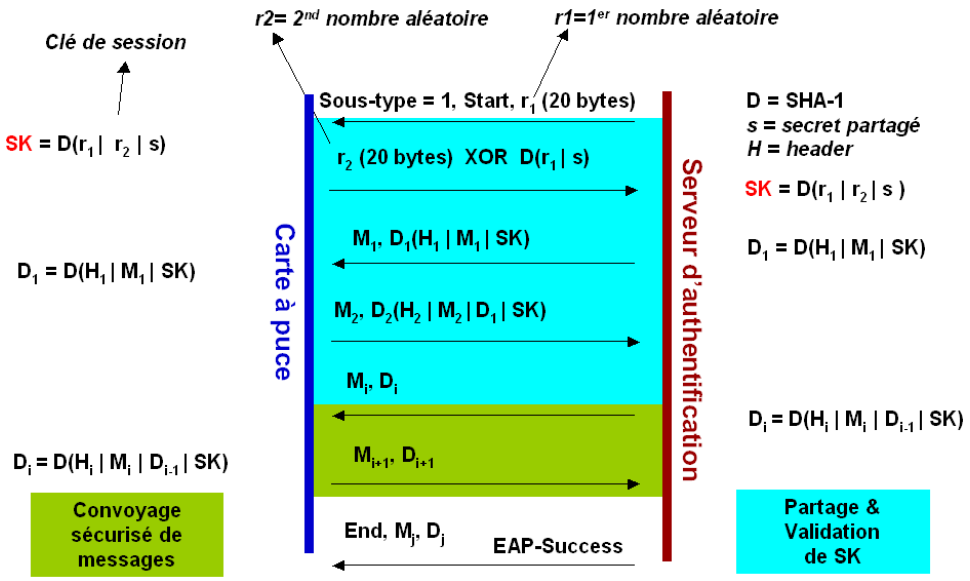


figure 5-3 : sous-type 1 associé au contexte de chiffrement à clés symétriques.

II.3.1. Phase de partage de clé de session EAP-SSC dans un contexte cryptographique à clés symétriques

Lorsque le serveur d'authentification reçoit de l'authentifiant un paquet *EAP-Response/Identity* valide, il génère un nombre entier aléatoire positif r1 de 160 bits de longueur (le bit de plus fort poids de r1 étant toujours à 0). Ce nombre r1 est empaqueté en clair, les octets de poids faible d'abord, dans un bloc *EAPOR-Request/EAP-SSC/Start* et envoyé à destination du demandeur d'accès à authentifier.

L'authentifiant qui reçoit le paquet *EAPOR-Request/EAP-SSC/Start* agit comme un relais qui, en fonction du format des paquets échangés entre lui et le demandeur d'accès, envoie en résultat à ce dernier un paquet *EAPOL-Request/EAP-SSC/Start*. A la réception de ce paquet, le demandeur d'accès à authentifier extrait l'entier positif aléatoire r1 envoyé par le serveur d'authentification.

Tout comme le serveur d'authentification, le demandeur d'accès lors de la préparation de sa réponse, génère aléatoirement un nombre entier positif r2 de 160 bits de longueur (avec ici aussi le bit de plus fort poids de r2 à 0) qu'il emploiera ensuite dans le calcul de la valeur Z=(r2 XOR SHA-1(r1 | s)), où s correspond au secret partagé par le serveur d'authentification et le client à authentifier, valeur qui dans le cas du demandeur d'accès est supposée conservée au niveau de la carte à puce. La valeur Z correspond en fait

au résultat du chiffrement du nombre aléatoire r2 que vient de générer le demandeur d'accès. Ce nombre Z est à son tour mis, les octets de poids faible d'abord, dans un paquet *EAPOL-Response/EAP-SSC* transmis au serveur d'authentification via l'authentifiant. A partir de cet instant, le demandeur d'accès à authentifier est en mesure de calculer la clé de session SK=SHA-1(r1 | r2 | s) du fait de la possession du triplet (r1, r2, s).

Lorsque le paquet *EAPOR-Response/EAP-SSC* aboutit au serveur d'authentification, celui-ci y récupère la valeur de Z. Du fait de la connaissance des valeurs de r1 et s, le serveur d'authentification est en mesure de calculer le condensé SHA-1(r1 | s), ce qui lui ouvre la voie au décryptage de la valeur de r2 cachée dans Z. Dès lors, le serveur d'authentification à son tour dispose du triplet (r1, r2, s) qui lui permet de calculer à son niveau la clé de session SK, tout comme le demandeur d'accès : on est ainsi au terme de la phase de calcul de la clé de session au niveau du protocole EAP-SSC dans le contexte cryptographique symétrique. L'étape de validation de cette clé de session pourra alors démarrer, ce que nous aborderons un peu plus loin après la présentation de la phase de partage de clé de session EAP-SSC dans un contexte cryptographique de clés asymétriques.

II.3.2. Phase de partage de clé de session EAP-SSC dans un contexte cryptographique à clés asymétriques

Dans le contexte cryptographique à clé publique, on fait plutôt usage de certificats qui sont des documents électroniques renfermant des clés publiques et d'autres données complémentaires permettant la réalisation des opérations de chiffrement et de déchiffrement.

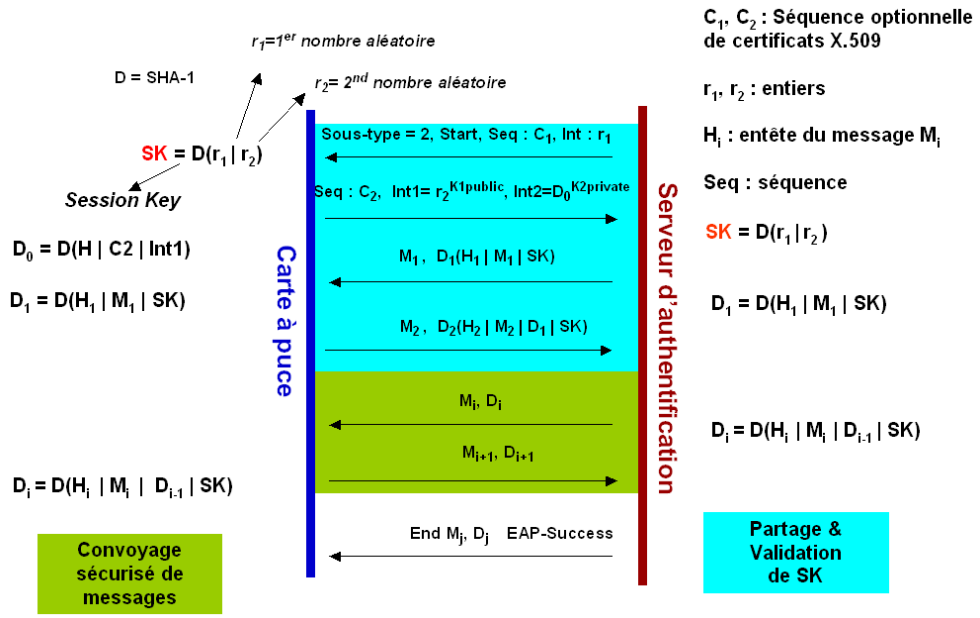


figure 5-4 : sous-type 2 associé au contexte de chiffrement à clés asymétriques.

Quand le serveur d'authentification dispose du certificat du demandeur d'accès et que ce dernier dispose aussi du certificat du serveur d'authentification, l'échange des

certificats peut être court-circuité ; et dans ce cas les paramètres C1 et C2 désignant les certificats échangés sont absents.

La phase de partage de la clé de session dans le contexte asymétrique démarre du côté du serveur d'authentification par la génération aléatoire d'un entier positif r1 de longueur variable, et dont l'octet de plus fort poids est à 0. r1 et la séquence optionnelle de certificats nommée C1 appartenant au serveur d'authentification sont représentés au format BER ASN.1 [ASN1_02] et envoyés en clair dans un paquet *EAPOR-Request/EAP-SSC/Start* expédié au demandeur d'accès via l'authentifiant. Dès que le demandeur d'accès va disposer du contenu du paquet *EAP-SSC/start*, il en extrait la valeur aléatoire positive r1 ainsi que la chaîne optionnelle de certificats C1. Par conséquent, il est en possession de la clé publique désignée par K1public d'une part, et de la base du modulo appelée Modulo1 de l'expéditeur du paquet reçu d'autre part.

Le demandeur d'accès, en réponse au message reçu, génère à son tour un nombre aléatoire positif r2 de longueur variable (mais avec toujours l'octet de plus fort poids à 0) dont il se sert pour calculer deux valeurs U et V telles que :

U=((r2 ** K1public) MOD Modulo1), et

V=((D0 ** K2private) MOD Modulo2), avec

D0=SHA-1(Code | Identifier | Length | Type | Sub-Type | Flags | C2 | U).

En fait D0 correspond au condensé calculé sur la concaténation de l'ensemble des champs précédant le champ *Digest* dans le paquet que le demandeur d'accès se prépare à envoyer en réponse au serveur d'authentification. La valeur de U correspond au chiffrement du nombre entier aléatoire r2 avec la clé publique du serveur d'authentification, tandis que celle de V correspond au cryptage du condensé D0 avec la clé privée du client à authentifier. C'est au format BER ASN.1 que la séquence optionnelle de certificats C2 et les nombres U et V sont rangés dans un paquet *EAPOL-Response/EAP-SSC* acheminé au serveur d'authentification via l'authentifiant. Dès lors, le demandeur d'accès est en mesure de calculer la clé de session SK=SHA-1(r1 | r2).

Lorsque le paquet *EAPOR-Response/EAP-SSC* parvient au serveur d'authentification, celui-ci en retire la séquence optionnelle des certificats C2 à partir desquels il retrouve la clé publique dénommée K2public et la base du modulo appelée Modulo2 employées par l'expéditeur du paquet. Grâce à la connaissance de ces deux valeurs, le serveur d'authentification est à son tour en mesure de déchiffrer la valeur D0 reçue et de la comparer à celle qu'il peut lui-même calculer à partir du paquet reçu. Si le condensé calculé localement et celui envoyé par le client à authentifier sont identiques, alors le serveur d'authentification poursuit avec la récupération du nombre aléatoire r2 à partir de la valeur de U, de sa clé privée K1private et de la valeur de base du modulo Modulo1. Autrement, le serveur d'authentification procède à la destruction silencieuse du paquet reçu. En possession du couple (r1, r2), le serveur d'authentification est lui aussi en mesure de calculer la clé de session SK=SHA-1(r1 | r2), mettant ainsi un terme à cette phase de partage de clé.

II.3.3. Phase de validation de la clé de session EAP-SSC

Succédant immédiatement à la phase de partage de la clé de session entre le serveur d'authentification et le demandeur d'accès, cette phase a pour principal objectif de vérifier que les deux interlocuteurs ont calculé la même valeur de clé de session SK, et de la sorte confirmer ou infirmer la mise en place d'un canal sécurisé d'échanges entre le demandeur d'accès et le serveur d'authentification. A l'initiative du serveur d'authentification, un

paquet *EAPOR-Request/EAP-SSC* est envoyé via l'authentifiant au demandeur d'accès à authentifier. La particularité de ce paquet est de renfermer un message désigné par M1 pouvant même être vide, mais systématiquement complété par un condensé de message D1=SHA-1(Header(M1) | M1 | SK), avec Header(Mi) correspondant à l'en-tête d'un paquet donné contenant le message Mi, et constitué des champs précédant le champ *Payload* (voir figure 5-2) dans le paquet. Dans le cas présent où le paquet achemine le message M1, l'en-tête Header(M1) correspond à la concaténation des champs *Code*, *Identifier*, *Length*, *Type*, *Sub-Type* et *Flags*.

A la réception de ce paquet qui a transité par l'authentifiant, le demandeur d'accès en extrait le condensé du message D1 provenant du serveur d'authentification. Il procède ensuite à la comparaison du condensé D1 reçu avec celui qu'il est en mesure de calculer lui-même car étant détenteur de la clé de session SK ; si elle échoue, alors le paquet sera silencieusement détruit. Autrement, on conclut à l'authentification correcte du serveur d'authentification par le client à authentifier qui, à son tour, calculera un nouveau condensé de message D2 selon la formule D2=SHA-1(Header(M2) | M2 | D1 | SK), où M2 correspond au message (pouvant être de longueur nulle) renvoyé en réponse à M1. M2 et D2 sont rangés dans un paquet *EAPOL-Response/EAP-SSC* expédié ensuite à destination du serveur d'authentification.

Quand finalement le paquet *EAPOR-Response/EAP-SSC* est reçu par le serveur d'authentification, ce dernier y récupère le condensé D2 (reçu). Disposant de D1 et de SK, le serveur d'authentification calcule localement à son tour une valeur de D2 et la compare au condensé D2 reçu. Dans le cas où ces deux condensés sont différents, le serveur d'authentification procède à la destruction silencieuse du paquet. Dans le cas contraire, on conclut à l'authentification correcte du client à authentifier par le serveur d'authentification. Cette fin de phase de validation termine l'authentification mutuelle entre le client à authentifier et le serveur d'authentification.

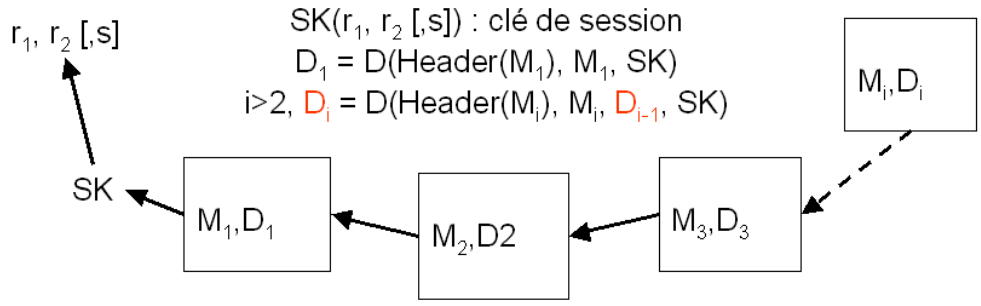


figure 5-5 : succession de condensés attachés aux messages échangés dans EAP-SSC.

II.3.4. Echange sécurisé de messages EAP-SSC

Dès la fin de la phase de partage de la clé de session SK, à chaque message échangé entre le serveur d'authentification et le demandeur d'accès est joint un condensé de message Di calculé à partir de la clé de session SK, du message Mi à expédier et du condensé Di-1 du message reçu à la précédente étape : D1=SHA-1(Header(M1)|M1|SK) et Di=SHA-1(Header(Mi) | Mi | Di-1 | SK) pour i > 1, avec SK calculé à partir soit du triplet (r1, r2, s) dans un contexte cryptographique à clé secrète, soit du couple (r1, r2) quand on se retrouve dans l'environnement cryptographique asymétrique. La figure 5-5 est une

illustration de l'enchaînement des condensés suffixés aux messages échangés lors d'une session d'authentification par le protocole EAP-SSC.

Au minimum trois messages M_1, M_2 et M_f (f égale à 3 ici) sont échangés :

- M_1 associé à une requête du serveur d'authentification et pouvant être de taille nulle ;
- M_2 correspondant à la réponse en provenance du demandeur d'accès à authentifier et pouvant être de longueur nulle ;
- M_f constituant le message final provenant du serveur d'authentification et chargé de mettre fin à l'échange sur le canal sécurisé. Le paquet EAP-SSC renfermant ce dernier message a le bit E (*End*) du *Flags* à 1. Comme à ce dernier message est aussi joint un condensé, tous les paquets imitant une fin de session n'auront aucun effet sur le comportement du demandeur d'accès. Entre les messages M_2 et M_f le serveur d'authentification et le client à authentifier peuvent continuer leurs échanges, avec joint à chaque message M_i le condensé de message D_i correspondant.

II.4. Chiffrement par défaut au sein du protocole EAP-SSC

L'un des meilleurs moyens dont on se sert pour assurer la confidentialité des messages échangés demeure le chiffrement. Au sein du protocole EAP-SSC l'utilisation du chiffrement des messages est indiquée par le positionnement du bit C du champ *Flags*. Par défaut l'algorithme de chiffrement employé est le 3-DES (*Triple Data Encryption Algorithm*) [DES99] [DES80] fonctionnant selon le schéma EDE (*Encode-Decode-Encode*) en mode CBC (*Cipher Block Chaining*) avec l'option 2 de clés (K1, K2, K3) où K1 et K2 sont des clés indépendantes tandis que K3=K1.

Nous fournissons en annexe II les résultats de simulation du protocole EAP-SSC à partir d'un jeu d'essai pour chacun des contextes cryptographiques.

III. Validation fonctionnelle du protocole EAP-SSC

EAP-SSC se base effectivement sur EAP et le 802.1X. Sa présence ne nécessite aucune modification de protocoles existants ; seule l'attribution d'une nouvelle valeur pour le type EAP par l'IANA est nécessaire.

Dans chacun des contextes cryptographiques, le protocole EAP-SSC réalise une authentification mutuelle qui est effective après la phase de validation de la clé de session SK.

La présence du bit C dans l'octet *Flags* permet d'indiquer le chiffrement de la charge utile transportée par un message EAP-SSC, un chiffrement qui est réalisé par défaut avec l'algorithme 3DES.

La garantie de l'intégrité des messages est simplement assurée par la présence d'un condensé calculé par SHA-1 et présent dans tous les messages échangés après la phase de validation de la clé de session.

La protection contre le rejeu est assurée grâce au calcul du condensé accompagnant chaque message et dans lequel on tient compte du condensé associé au message antérieurement envoyé. Donc si l'attaquant n'a pas tous les condensés d'une session, il lui est impossible de rejouer des séquences de messages EAP-SSC. De plus, la clé de session SK est générée dynamiquement pour chaque session à partir des nombres aléatoires fournis par le demandeur d'accès et le serveur d'authentification, complétés de la clé secrète partagée quand on est dans un contexte cryptographique symétrique.

Toutes les opérations réalisées se résument à des exponentiations, des calculs de modulo et des OU exclusifs. Nous en avons fait une simulation dans l'environnement JBUILDER 7. Nous pensons qu'au sein d'une carte à puce disposant d'un co-processeur cryptographique, elles pourraient être faites sans difficulté.

Dans le cas du contexte de chiffrement symétrique, on suppose que le demandeur d'accès et le serveur d'authentification disposent de la clé secrète en partage. Aucun transport de cette clé sur le support hertzien n'est prévu. Les flux de données échangées se limitent en gros à ceux de deux nombres aléatoires.

Dans le cas d'une authentification dans le contexte asymétrique, la possession par chaque entité du certificat de l'autre partie présente des similitudes avec l'authentification dans le contexte symétrique. Seulement, ici la sécurité par rapport aux attaques de l'homme du milieu est plus faible. En effet, si un attaquant "homme du milieu" dispose des clés publiques du serveur d'authentification et du demandeur d'accès et fait partie du même cercle de confiance qu'eux, il peut agir sans que l'on s'en aperçoive. C'est d'ailleurs pourquoi, la circulation des certificats non chiffrés sur le support radio peut poser des problèmes.

IV. Faiblesses du protocole EAP-SSC

Schneier [SCH01] rappelle que « *L'analyse cryptographique est une discipline complexe et les personnes compétentes sont rares. Pour qu'une primitive puisse être considérée comme sûre, elle doit être examinée par de nombreux experts au fil des ans. C'est pourquoi les cryptographes préfèrent ce qui est ancien et public à ce qui est nouveau et propriétaire. La cryptographie publique est étudiée par les cryptographes et fait l'objet de publications. Les primitives les plus anciennes sont les plus largement décrites dans des articles. Si elles présentaient des défauts, on les aurait déjà trouvés (c'est en tout cas l'argument avancé). Ce qui est nouveau est plus risqué, précisément parce c'est nouveau et que trop peu de personnes l'ont étudié.*» Le draft proposé n'a été soutenu que durant deux périodes successives de 6 mois.

En effet, le protocole EAP-SSC a été voulu comme une façon de se satisfaire des capacités des cartes à puce du moment (2003), sans tenir compte de la loi de Moore, de l'expertise et du temps qu'il faut pour en faire un protocole de sécurité. Or, une année plus tard, notre équipe de recherche réussissait à implémenter au sein d'une carte à puce Java le protocole EAP/TLS de notoriété mondiale [EAPTLS04]. Une compétition est alors née entre EAP/TLS et EAP-SSC, de laquelle est sortie gagnante EAP/TLS qui correspondait au choix de l'IETF.

Le protocole EAP-SSC n'a pas été conçu pour offrir un choix de la suite de chiffrement à employer lors d'une session d'authentification. Toute modification de la suite de chiffrement peut être considérée comme la définition d'un nouveau scénario d'authentification qui pourrait être pris en compte dans le champ *Sub-Type*. Le bit R (*reserved*) dans le champs *Flags* par défaut à 0 pourrait aussi servir à étendre le champ *Flags*.

Dans le cadre de l'amélioration de ce protocole, on peut imaginer le renforcement du contrôle d'intégrité des messages échangés en employant un HMAC-SHA-1 calculé avec la clé de session plutôt que d'utiliser un simple condensé.

Dans ce protocole, il n'est pas non plus prévu de gérer une quelconque liste de révocation de certificats, ou encore de réaliser des vérifications en ligne par le protocole OCSP (*Online Certificate Status Protocol*) [OCSP99] de la qualité des certificats manipulés.

Par rapport aux nombres aléatoires utilisés, il est recommandé une génération particulière de nombres aléatoires ou pseudo-aléatoires qui aboutissent à la production de clés fortes (différentes des clés faibles qui fragilisent les algorithmes de chiffrement). De même, il est conseillé que la taille en bits de ces nombres générés aléatoirement soit au moins égale à celle des blocs employés dans les algorithmes de hachage ; par exemple dans le cas du protocole EAP-SSC employant SHA-1, les nombres aléatoires devraient au moins avoir une taille de 512 bits (64 octets) au minimum.

V. Comparaison des protocoles EAP-SSC et EAP-TLS

Afin de bien montrer les traits de différence et de ressemblance entre les protocoles EAP-SSC et EAP-TLS, nous dressons dans le tableau 5-1 une comparaison des deux protocoles.

tableau 5-1 : comparaison des protocoles EAP-SSC et EAP-TLS.

	EAP-SSC (2003)	**EAP-TLS (2004)**
Modes cryptographiques	• Symétrique • Asymétrique	Asymétrique avec un mode *resume* symétrique
Echange de certificats en mode asymétrique	Optionnel	Obligatoire
Négociation de suites cryptographiques	NON (3DES, SHA-1)	OUI
Nombre de passes	4	6
Taille des échanges lors d'une session d'authentification	200 – 1800 octets	2600 octets, avec 800 octets par certificat et une clé de 1024 bits
Difficultés d'implémentation dans la carte à puce (2003)	Aucune car orienté carte à puce	Grandes

EAP-SSC, pour faire simple, n'autorise pas de négociation de suites cryptographiques ; les outils cryptographiques reconnus par la communauté scientifique comme le triple DES et le SHA-1 sont employés. Quatre questions/réponses sont utilisées lors de l'authentification dans le protocole EAP-SSC, au lieu de 6 au niveau de EAP-TLS. Si en mode de chiffrement symétrique, le volume de données échangées lors du déroulement du protocole EAP-SSC est moins du 10^{e} de celui nécessaire au protocole EAP-TLS "*mode resume*", en mode de chiffrement asymétrique ce rapport est un peu plus de moitié.

VI. Conclusion

A travers ce chapitre, nous avons montré les détails du fonctionnement du protocole d'authentification EAP-SSC. Il ressemble à une méthode EAP-TLS simplifiée, en ce sens qu'il réalise une authentification mutuelle sans négociation des suites de chiffrement, et utilisable dans les deux contextes cryptographiques. Du fait des outils mathématiques basiques sur lesquels se fonde son chiffrement, le protocole EAP-SSC s'autorise dans l'environnement de la carte à puce EAP. Nous aurions continué l'étude de ce protocole si d'une part en application de la loi de Moore la carte à puce hébergeant le scénario EAP/TLS n'avait été réalisée, et d'autre part si l'intérêt de la communauté des cartes pour ce protocole avait été plus marqué.

Remarquons que l'idée d'un protocole d'authentification mutuelle utilisable dans les deux contextes de chiffrement a été reprise en 2005, à travers le draft IETF intitulé EAP-PAX [EAPAX05]. Il propose deux sous-protocoles d'authentification, le premier PAX-STD reposant sur des clés symétriques et l'autre plus robuste dénommé PAX-SEC s'appuyant sur une infrastructure de clés publiques PKI (*Public Key Infrastructure*). Dans ce protocole l'échange de germes aléatoires est fait par le moyen de l'algorithme de Diffie-Hellman [PKCS3_93] [DIFHEL03], et un service de distribution de clés est également proposé.

Comme si le mode "*resume*" de EAP-TLS restait insatisfaisant, l'IETF vient de normaliser en fin décembre 2005 le protocole TLS-PSK (*Transport Layer Security Pre-Shared Key*) [TLSPSK05] qui n'est autre chose que du TLS en mode de chiffrement symétrique.

Chapitre 6 : La plate-forme *OpenEapSmartcard*

Si en 2005 l'on dénombrait environ un milliard et demi d'abonnés du GSM dans le monde, l'essor actuel des réseaux sans fil IP nous amène à penser que dans quelques années nous serons plongés dans des constellations de réseaux sans fil IP occupant tous nos lieux de vie (domicile, transport, travail, loisirs). Il se posera alors le problème de l'administration et de la sécurité de tous ces réseaux.

Tout comme la présence de la carte SIM au sein du réseau GSM a permis d'atteindre les objectifs de sécurité et de mobilité, nous pensons qu'il en sera de même pour les réseaux sans fil IP avec l'emploi de la carte à puce EAP comme support de sécurité. Or à la différence des réseaux GSM, les réseaux sans fil sont sous diverses autorités administratives. Dans l'objectif de faciliter le déploiement d'une diversité de solutions sécurisant l'accès aux réseaux sans fil IP, nous présentons la première architecture ouverte implémentant le protocole EAP dans les cartes Java.

Le chapitre débute avec la présentation des objectifs et des chances de l'initiative *OpenEapSmartcard*. Nous dressons ensuite une liste des difficultés rencontrées au cours de la réalisation de la plate-forme *OpenEapSmartcard*. Le cœur du chapitre concerne l'architecture retenue pour elle. Les résultats des tests de performances effectués entre des cartes de différents fabricants d'une part, et déroulant diverses méthodes d'authentification d'autre part, viennent clore ce chapitre. Cette plate-forme a fait l'objet des publications [OpnEap05] et [UrDa05].

I. Objectifs et chances du projet *OpenEapSmartcard*

L'objectif du projet *OpenEapSmartcard* est de mettre en place une plate-forme EAP Java ouverte, simple et transparente [OpnEap05] [UrDa05] [OpnEap06], disposant de l'interface avec les APDU décrite dans [EAPSup06], fonctionnant avec la plupart des cartes Java du marché, et supportant le plus de méthodes EAP possibles.

Nous définissons l'ouverture de notre plate-forme comme étant le fait de rendre accessible à la communauté des développeurs le contenu logiciel de la plate-forme. Cette notion d'ouverture porte également en elle la capacité d'adaptation de la plate-forme à différents environnements et d'en vérifier le fonctionnement. Elle constitue un critère d'obtention de la sécurité, comme le sont bon nombre d'algorithmes réputés sûrs que la cryptographie met à la disposition des utilisateurs. Schneier [SCH01] ne rappelle-t-il pas que "*pour qu'une primitive puisse être considérée comme sûre, elle doit être examinée par de nombreux experts au fil des ans. C'est pourquoi les cryptographes préfèrent ce qui est ancien et public à ce qui est nouveau et propriétaire. La cryptographie publique est étudiée par les cryptographes et fait l'objet de publications*". La plate-forme *OpenEapSmartcard* va de la sorte contre l'idée d'employer des "boîtes noires" pour faire de la sécurité. Et les nombreuses failles de sécurité découvertes dans le cas du GSM (le crack COMP 128-1) ou du 802.11 (le crack du WEP) nous poussent à investir dans un code ouvert/libre offrant plus de transparence.

Quant à la notion de transparence, nous l'opposons à celle dissimulation. En effet, la mise à disposition du code source Java de la plate-forme EAP devrait permettre de poser les bases du développement d'une bibliothèque ouverte de scénarii qui ne pourra que s'enrichir avec le temps. Les contributions seront mondiales, provenant aussi bien des fondeurs, des équipementiers, des éditeurs de logiciels, des laboratoires de recherche que de particuliers. Des tests et vérifications en tout genre assureront les améliorations

nécessaires au jour le jour pour garantir la sécurité qui par définition n'est jamais acquise une fois pour toutes.

De façon indirecte, le projet devrait pousser vers le haut la qualité des produits de l'industrie de la carte à puce en créant une véritable émulation. En effet, peu de publications existent dans le domaine de la comparaison des performances des différentes cartes. Cette plate-forme serait le cadre désigné pour la conduite de tests comparatifs.

Pour atteindre ces objectifs, nous pensons disposer d'un certain nombre d'atouts qui sont principalement :

- L'utilisation de l'environnement d'exécution Java qui est reconnu pour sa sécurité, son ouverture et sa portabilité.
- La résistance des cartes à puce à la falsification, ce qui assure à son porteur tout comme à son émetteur une sécurité. La protection de premier niveau au moyen du code PIN pourra bientôt être améliorée par l'usage des données biométriques. Difficile d'en faire une copie si l'on ne dispose pas des privilèges de son émetteur/administrateur.
- La grande vague d'adoption de la technologie du GSM représente pour nous un bon signe car si la carte SIM a permis de sécuriser l'accès au réseau GSM, une carte à puce Java EAP en fera autant pour les réseaux sans fil IP ; et ainsi assisterait-on à sa grande utilisation à travers le monde.
- Le faible coût du matériel requis pour notre solution contribue à la rendre plus attrayante. Par exemple en 2006, on peut trouver sur le marché des cartes à puce Java à moins de 10 euros l'unité, et des lecteurs USB de cartes à puce à moins de 40 euros la pièce.
- La multiplicité des formes de présentation des cartes à puce, allant des cartes de crédits habituelles aux cartes et token d'interface USB, facilite le transport et l'utilisation de la plate-forme EAP logée en leur sein.
- La poussée technologique permanente (loi de Moore) qui permet de repousser encore les limites de la mémoire de stockage et de la puissance de traitement. Il existe de nos jours des cartes à puce capables d'effectuer des calculs de clés RSA 2048 bits en moins de 500 ms avec une mémoire de taille voisine de 64 ko. Qu'en sera-t-il avec les mémoires flash d’un méga-octet qui commencent à être commercialisées ?

II. Exigences à prendre en compte

La première exigence de l'initiative *OpenEapSmarcard* est la modularité qu'elle doit proposer et maintenir. En effet, un module disponible peut être rapidement intégré et évalué, et ainsi faciliter l'étude de nombreuses autres architectures de sécurité. De plus, on peut correctement estimer les coûts de déploiement induits par l'utilisation des cartes à puce ou la gestion de l'infrastructure qu'elle suppose (gestion de la clé secrète , gestion des certificats, sûreté des communications, etc.).

La deuxième exigence est la faisabilité de la solution déjà évoquée à la fin du chapitre portant sur "le protocole EAP et la carte à puce". Comme une authentification est une séquence de requêtes envoyées au client (la carte à puce dans notre proposition) qui, à son tour, produit des réponses, et que selon [802.1X_01] le temps par défaut associé au RTT (*Round Trip Time*) doit être inférieur à 30 s, il y a une contrainte sur le temps d'authentification qui doit être en dessous de cette limite.

Une dernière contrainte est celle de la complexité des protocoles d'authentification dont la taille du code source est limitée par la capacité de la mémoire non volatile généralement comprise entre 32 et 128 ko ; heureusement que cette taille pourrait rapidement évoluer vers le méga-octet. Des méthodes complexes d'authentification telles que EAP-TLS [EAPTLS99] ou EAP-PSK [EAPPSK04] qui ont besoin d'au moins 20 ko de mémoire non volatile devraient pouvoir marcher avec un temps RTT de moins de 30 secondes.

III. Architecture de la plate-forme *OpenEapSmartcard*

III.1. Présentation générale

La plate-forme *OpenEapSmartcard* [OpnEap05] [UrDa05] est dédiée aux environnements carte à puce Java (JC), bâtis autour d'un micro-contrôleur (CPU de 8 à 32 bits) disposant au moins d'une ROM de 64 ko, d'une mémoire non volatile de 32 ko (EEPROM, Flash) et de 4 ko de mémoire vive RAM. En accord avec la caractéristique intéressante de la technologie Java qu'exprime le slogan "*Write Once, Run anywhere …*", cette plate-forme devrait fonctionner avec tous les dispositifs supportant les API JC2.1 ou JC2.2 [CHEN00]. Elle s'appuie à la base sur un ensemble de commandes ISO/IEC 7816 dont les détails sont fournis dans le draft IETF [EAPSup06].

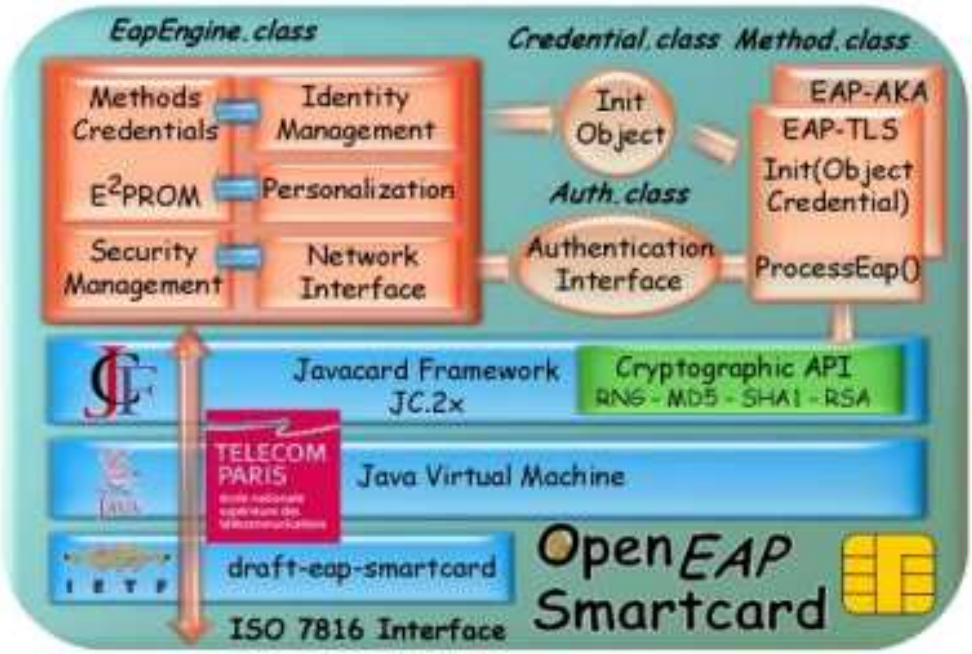


figure 6-1 : plate-forme *OpenEapSmartcard*.

Dans l'architecture logicielle de la plate-forme *OpenEapSmartcard* se distinguent principalement quatre composants :

- Le *moteur EAP* qui est chargé de la gestion de plusieurs méthodes et/ou de multiples instances de la même méthode.
- L'*interface d'authentification* qui définit tous les services obligatoires des méthodes EAP, de sorte à pouvoir travailler en relation avec le moteur EAP.

- Les *lettres de crédits* ou "*credentials*" qui vont chacune être associées à une méthode donnée, et renfermer toute l'information nécessaire à la réalisation d'un scénario d'authentification spécifique.
- Les *méthodes* qui correspondent aux scénarii spécifiques d'authentification à traiter. Une fois initialisée, la méthode retenue analyse chaque requête EAP entrant et délivre la réponse correspondante.

III.1.1. Le moteur EAP

C'est la classe d'objet *EapEngine* qui implémente le moteur EAP (figure 6-1). Il correspond au noyau du protocole EAP, et se comporte comme un routeur envoyant et recevant des paquets EAP à destination et en provenance des méthodes d'authentification.

Quatre services sont rendus par ce moteur, chacun correspondant à un bloc fonctionnel davantage détaillé dans [BaUr04] :

- L'*interface réseau*. Les messages EAP-Request en entrée sont vérifiés et transmis à la méthode concernée. C'est elle aussi qui en retour reçoit les messages renvoyés par les méthodes. A la fin d'un processus d'authentification réussie, chaque méthode est supposée avoir calculé une clé cryptographique principale dite clé AAA mise à la disposition du système d'exploitation du terminal hôte.
- La *gestion de l'identité*. La carte à puce gère plusieurs méthodes et/ou de nombreuses instances de la même méthode, chacune associée à des lettres de crédits. Pour une gestion aisée de ces lettres de crédits, on associe à chaque méthode une *identité*, véritable clé primaire dans une base de données. Par exemple en rapport avec la méthode EAP-TLS, on a une identité qui permet de retrouver les lettres de crédits formées par l'identificateur de session EAP-ID, le certificat du client, la clé privée du client, et la chaîne de certificats X.509 habituellement restreinte à une autorité de certification primaire. Ce service permet de consulter les identités disponibles et de choisir l'une d'entre elles. Plusieurs identités peuvent être activées au même moment, associées à plusieurs sessions EAP, et employées dans différents contextes comme celui du *Supplicant* dans l'architecture IEEE 802.1X ou celui d'une négociation IPsec [IPSEC98] avec le protocole IKEv2 [IKEv2_05].
- La *gestion de la sécurité*. La carte à puce est protégée par deux codes PIN, l'un pour son émetteur et l'autre pour son propriétaire. Dans les versions futures, ce service pourrait connaître des améliorations avec la prise en compte des données biométriques.
- La *gestion de la personnalisation*. Les éléments d'identité et les codes PIN sont contrôlés et mis en place par les émetteurs de cartes à puce. Le canal sécurisé créé à la suite du bon déroulement de certaines méthodes d'authentification comme EAP-TLS, pourrait convenir dans le cadre d'une administration à distance de la puce, à la mise à jour *Over The Air* de certains paramètres. Or les opérations de cette nature nécessitent qu'on puisse faire appel à un module de personnalisation. Ce modèle ressemble au standard qui est largement déployé dans le réseau GSM [GSM03.48] pour modifier le contenu de certains fichiers de la carte SIM.

III.1.2. L'interface d'authentification

Sa présence est associée à la classe *Authentication Interface* (Interface d'authentification) qui, selon le paradigme objet classique, décrit les services permettant de

collaborer avec le moteur EAP (tableau 6-1). Deux services principaux y sont définis : *Init()* et *Process-EAP()*.

Init() a en charge l'initialisation ou la réinitialisation d'une méthode. Ce service consiste d'une part à envoyer à la méthode qui les emploie les lettres de crédits (identité, clés, références des ressources cryptographiques, etc.) encapsulées dans un objet Java générique, et d'autre part à retourner une interface d'authentification.

Process-EAP() quant à lui traite les paquets EAP entrants en fonction de la méthode retenue. D'autres méthodes pourraient venir enrichir cette interface de services complémentaires comme par exemple ceux dédiés à l'évaluation des performances d'une méthode.

tableau 6-1 : interface d'authentification.

Interface auth	
void	fct(javacard.framework.APDU apdu, byte[] in, short inlength) fonctions de la méthode apdu : APDU entrant in : tampon associé à l'APDU entrant inlength : valeur de type P3 caractérisant la taille des données en cause
byte[]	Get_Fct_Buffer() retourne le tampon d'une fonction
short	Get_Fct_Length() retourne la taille du tampon d'une fonction
short	Get_Fct_Offset() retourne la position relative (offset) du tampon d'une fonction
byte[]	Get_Out_Buffer() retourne le tampon de la réponse
short	Get_Out_Length() retourne la taille du tampon de la réponse
short	Get_Out_Offset() retourne la position relative (offset) du tampon de la réponse
auth	Init(java.lang.Object credentials) Initialisation d'une méthode
boolean	IsFragmented() signifie que la fragmentation est en cours
boolean	IsLongFct() signifie que la réponse d'une fonction est conservée dans un tampon privé
boolean	IsLongResponse() signifie que la réponse de la fonction est de type long (c'est-à-dire pouvant atteindre 65535 octets)
short	process_eap(byte[] in, short inlength) réalise la méthode In : tampon de l'APDU entrant Inlength : taille de l'APDU entrant Cette méthode retourne : • une valeur positive correspondant à la taille de la réponse • une valeur négative dans le cas d'une erreur
void	reset() pour réinitialiser la méthode
short	status() pour retourner l'état (le statut) de la méthode

III.1.3. Les lettres de crédits

Chaque méthode est associée à des lettres de crédits spécifiques renfermant toutes les informations requises pour la réalisation du scénario d'authentification correspondant. C'est la classe *Credential_Objects* qui l'implémente. Une instance de cette classe est initialisée et poussée vers la méthode associée lorsque l'identité est sélectionnée par le moteur *EapEngine*.

III.1.4. Les méthodes

A chaque scénario d'authentification correspond une méthode particulière. Il y a autant de méthodes différentes que de classes *Methods* implémentées. Une fois initialisée, l'instance d'une méthode analyse chaque requête EAP qui entre, puis délivre la réponse correspondante. A la fin d'un scénario d'authentification réussie, l'instance de la méthode calcule une clé secrète. Les opérations de segmentation et de ré-assemblage des paquets EAP sont pour une part réalisées par les méthodes, et pour l'autre part confiées au moteur *EapEngine*. Le nombre de méthodes embarquées dans une carte à puce est limité par la taille de sa mémoire non volatile.

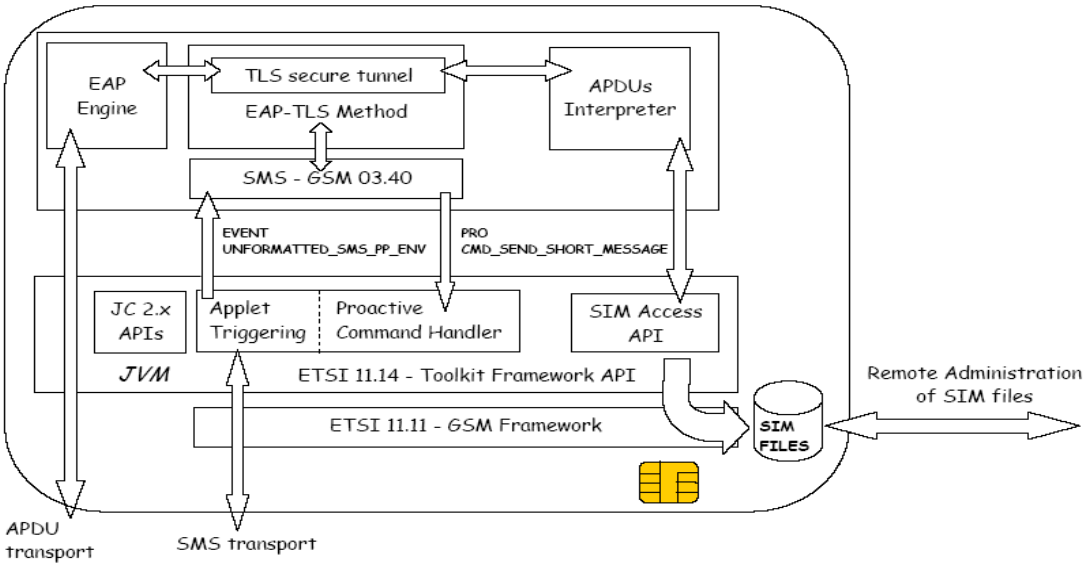


figure 6-2 : support *OpenEApSmartcard* dans les cartes SIM.

III.2. OpenEapSmarcard dans les cartes SIM

Du fait de la flexibilité du système fondé sur Java, nous avons procédé à l'intégration de la plate-forme *OpenEapSmartcard* dans une carte SIM comme l'illustre la figure 6-2. Dans notre construction faite avec une carte Java SIM classique, et à des fins de démonstration, nous réalisons un transport transparent des messages EAP-TLS à travers la charge utile des messages courts SMS. Une autre interface est disponible et traite les messages EAP-TLS transportés par les APDU [BaUr04].

Comme spécifié dans le standard [GSM03.40], les paquets SMS sont constitués d'un en-tête dont la taille est moins de 35 octets, et d'une charge utile de 140 octets maximum. Les méthodes EAP supportent un mécanisme de fragmentation, ce qui permet de fixer une taille maximale de paquet à 140 octets compatible avec les capacités du SMS. Puisque la plate-forme *OpenEapSmartcard* ne supporte pas le protocole de sécurité défini dans le GSM [GSM03.48], elle le remplace par EAP-TLS qui, de la même façon que procède le protocole de sécurité du contexte GSM, ouvre de manière flexible et extensible un tunnel sécurisé.

Etant donné que les cartes SIM classiques sont destinées au GSM dont la sécurité puise dans les ressources de la cryptographie à clés symétriques, elles ne supportent pas les mécanismes RSA ; c'est donc le mode *"TLS resume"* que nous avons fait fonctionner sur notre plate-forme SIM expérimentale à partir d'une ancienne clé secrète identifiée par un

paramètre Session-ID. Cependant, si nous disposions de ressources RSA, l'ouverture d'une session normale d'authentification aurait pu être envisagée.

En résumé, c'est donc deux canaux de transport qui sont disponibles pour l'administration de la carte SIM *OpenEapSmartcard* : les messages courts SMS et les APDU classiques. Le contenu des messages courts est d'une part protégé par un tunnel TLS, et d'autre part analysé par un bloc interpréteur d'APDU qui invoque les services de l'API *SIMAccess*.

III.3. Intégration aux terminaux

Certains systèmes d'exploitation comme Win32 supportent déjà le protocole EAP pour les réseaux filaires et sans fil. Un scénario d'authentification EAP donné est traité par une librairie dynamique (DLL) particulière appelée EAP-provider [UrLo03]. Une bibliothèque générique qui emploie les services PC/SC [PCSC96] (support *plug and play* des lecteurs de cartes à puce) transmet les requêtes EAP entrant à la carte à puce qui calcule les réponses correspondantes.

IV. Résultats expérimentaux et performances

Les premières réalisations de la plate-forme *OpenEapSmartcard* ont ciblé l'emploi de l'une des trois méthodes d'authentification suivantes : EAP-TLS, EAP-PSK et EAP-AKA. C'est dans cet environnement qu'ont été produits les résultats que nous présentons et qui ont fait l'objet d'une publication [UrDa05].

IV.1. Problèmes d'interopérabilité

Du fait de l'universalité du langage Java, l'espoir était que le même code puisse marcher avec toutes les cartes à puce ; la réalité est toute autre à cause des différences mineures et des bugs qui existent presque toujours entre les cartes. Nous listons brièvement quelques-uns des problèmes d'interopérabilité que nous avons rencontrés :

- Dans le protocole TLS, les calculs RSA sont réalisés en employant des règles de bourrage du PKCS#1 [PKCS1_02]. Quelquefois, cette fonctionnalité n'est pas disponible et seule l'option NO_PAD (absence de bourrage) marche ; il faut de ce fait recourir à du code Java additionnel.
- Les méthodes employées lors du calcul des condensés sont souvent *update()* pour la conservation des résultats temporaires de calcul, et *DoFinal()* pour la fin du traitement et la récupération du résultat terminal. Or parfois la méthode *update()* n'est pas disponible, et par conséquent l'on est obligé de concaténer d'abord l'ensemble des données en entrée de la méthode de calcul du condensé dans la mémoire non volatile avant de pouvoir faire le calcul en une seule fois.
- Il est arrivé de constater des valeurs erronées produites par la méthode *update()* utilisée avec un grand bloc d'octets (quelques centaines d'octets) en entrée. Ces erreurs de calcul pourraient être le résultat d'une mauvaise écriture du code de cette méthode.
- Il n'est parfois possible de travailler qu'avec une seule instance de MD5 [MD5_92] ou de SHA1 [SHA1_02]. Par conséquent, une application inter-opérable ne peut utiliser qu'une seule instance de ces méthodes, ce qui implique de nombreuses

écritures dans la mémoire non volatile, et partant une baisse des performances. On rappelle que TLS par exemple emploie trois calculs de MD5 et de SHA1.

La méthode EAP-TLS implémentée dans la première version de carte *OpenEapSmartcard* tient compte de ces problèmes : elle fonctionne avec un algorithme RSA n'employant aucun octet de bourrage ; elle est compatible avec une simple instance de calcul de condensé et gère les méthodes absentes ou contenant des bugs.

IV.2. Résultats avec la méthode EAP-TLS

Les tests ont été effectués sur quatre cartes à puce JC2.1 équipées chacune d'un processeur 8 bits battant à 8 MHz et provenant de différents fabricants (tableau 6-2). Elles sont toutes dotées d'une mémoire EEPROM de 32 ko, et d'une mémoire RAM dont la taille varie entre 500 et 1024 octets. Les équipements dénommés A, C et D sont des cartes Java généralistes tandis que l'équipement dénommé B est une carte SIM. Chacune de ses cartes a son propre environnement de développement.

tableau 6-2 : principales caractéristiques des cartes à puce employées.

Code	Nom (fabricant)	Caractéristiques générales
A	e-gate (Axalto)	• Processeur 8 bits battant à 8 Mhz • RAM : 500 – 1024 octets • EEPROM : 32 ko
B	SIM (Gieseke)	
C	Jcop (IBM)	
D	Gemplus (Gemplus)	

Les caractéristiques cryptographiques additionnelles qui ne sont pas supportées par l'environnement JC2.x sont écrits en Java, comme HMAC, RC4, TLS, PRF (*Pseudo Random Function*) et les analyseurs de certificats X.509.

IV.2.1. EAP-TLS "full mode"

De l'analyse du tableau 6-3 on peut tirer les premiers enseignements suivants :

- Trois cartes parmi les quatre testées ont un RTT inférieur à 30 s ; ce sont les cartes dénommées B, C et D. Elles pourraient donc servir pour les tests ultérieurs.
- Durant le scénario d'authentification environ 2500 octets ont au total fait l'objet d'un transfert (envoi et réception) par la carte à puce, ce qui consomme entre 2 et 5 s (voir colonne "*Data transfer*", tableau 6-3). On notera que ce temps est principalement fonction de la vitesse de transfert entre la puce et le lecteur de carte, des performances de la machine virtuelle Java embarquée et des temps de lecture/écriture des mémoires non volatiles des puces.
- Selon la spécification TLS, trois condensés doubles (MD5 + SHA1) sont calculés pour les données échangées, c'est-à-dire environ 7500 octets (près de 120 blocs de 512 bits). On déduit de la colonne "*Dual Hash*" du tableau 6-3 un temps moyen de calcul d'un double condensé par bloc de 512 bits compris entre 3,7 (900/240) ms et 24 (5700/240) ms.
- Cinq occurrences de la fonction pseudo-aléatoire (PRF) ont impliqué la réalisation de 31 signatures HMAC-MD5 et de 31 signatures HMAC-SHA1 qui ont traité 140

blocs MD5 et 140 blocs SHA1. Chaque HMAC a calculé deux condensés ayant approximativement une taille de 2,25 blocs. C'est à ce niveau qu'apparaît la plus forte consommation relative de temps comprise entre 40 et 60 % (colonne "*PRF*"), et imputable très probablement à l'implémentation additionnelle en Java.

- L'algorithme RC4 est aussi complètement écrit en Java, et les performances observées (entre 5 et 20 % du temps total consommé par l'authentification) illustrent clairement la variété des caractéristiques des différentes machines virtuelles embarquées.
- Entre 10 et 30 % du temps global sont employés au titre du surcoût de temps (*overhead*) imputable à la présence de la machine virtuelle Java.

tableau 6-3 : temps mesurés durant une session d'authentification TLS "*full mode*".

CODE	RTT Max (s)	Total Authen-tication Time (s)	Data Transfer 2500 bytes Time (s)	Dual Hash 2 x 120 blocs Time (s)	PRF 2 x 140 blocs Time (s)	RC4 2 x 32 bytes Time (s)	Other Time (s)
A	52,3	78,1	2,3 2,9%	5,7 7,3%	42,4 54,2%	14,7 18,9%	13,0 16,6%
B*	21,0	34,3	4,3 12,5%	4,5 13,2%	19,7 57,3%	2,3 6,7%	3,5 10,2%
C	22,3	33,3	4,9 14,7%	3,5 10,4%	13,3 40,0%	2,8 8,4%	8,8 26,5%
D	5,2	9,3	1,6 17,2%	0,9 9,7%	4,5 48,4%	0,7 7,5%	1,6 17,2%

*** : carte SIM**

Les ressources RSA mises à disposition par le processeur cryptographique sont employées trois fois avec des clés de 1024 bits. Si les opérations avec la clé publique nécessitent autour de 200 ms, celles utilisant la clé privée sont plus lentes et consomment près de 400 ms (voir tableau 6-4).

tableau 6-4 : performances RSA de quatre cartes à puce.

	RSA 1024 bits Public Key Initialization Time (s)	RSA 1024 bits Encryption Time (s)	RSA (Verify data) 1024 bits Private Key Encryption Time (s)
A	0,21	0,16	0,33
B*	-	-	< 0,80
C	0,31	0,38	0,43
D	-	0,02	0,11

* : carte SIM

IV.2.2. EAP-TLS *resume mode*

Ce mode de déroulement de la session d'authentification est caractérisé par l'échange de près de 250 octets, l'emploi d'une clé secrète antérieurement calculée, et aucun calcul RSA. Deux condensés doubles (MD5 + SHA1) sont calculés sur à peu près 2x150 octets (5 blocs de 512 bits). Quatre occurrences de la fonction PRF impliquent la réalisation de 15 signatures par la procédure HMAC-MD5 et 15 signatures par la procédure HMAC-SHA1 traitant 108 blocs MD5 et 108 blocs SHA1. Chaque HMAC calcule deux condensés dont la taille est approximativement de 3,6 blocs.

tableau 6-5 : temps mesurés lors d'une session d'authentification TLS en "*resume mode*".

CODE	Total Authentication Time (s)	Data Transfer 250 bytes Time (s)	Dual Hash 2 x 6 blocs Time (s)	PRF 2 x 108 blocs Time (s)	RC4 2 x 32 bytes Time (s)	Other Time (s)
A	49,5	0,9 1,9%	0,3 0,6%	32,5 65,6%	14,7 29,7%	1,1 2,2%
B*	18,7	1,0 5,4%	0,2 1,0%	15,2 81,3%	2,3 12,3%	0,0 0%
D	5,5	0,2 3,6%	0,0 0%	3,5 63,7%	0,7 12,7%	1,1 20,0%

NB : au moment de ces tests nous ne disposions plus de la carte de code C (Jcop).

Du tableau 6-5 ci-dessus il apparaît que 60 à 80 % du temps de la session sont consacrés au calcul du PRF, et entre 10 et 30% pour le chiffrement RC4. Et comme on pouvait s'y attendre, le mode abrégé est plus rapide (entre 5 et 50 s) que le mode complet (entre 10 et 80 s) ; il pourrait donc être utile dans le cas de la re-authentification rapide d'un client. 5 à 20 % du temps sont ici associés à l'*overhead*.

IV.3. Résultats avec la méthode EAP-PSK

Les résultats que nous présentons portent sur la méthode EAP-PSK [EAPPSK04] qui s'appuie sur l'algorithme AES [AES01]. La plupart des cartes Java actuelles implémentent le standard JC2.1 qui malheureusement ne supporte pas l'algorithme AES. En revanche cet algorithme est présent dans la version JC2.2 des cartes à puce. Aussi, nos tests se sont-ils réalisés en employant deux sortes d'instances AES, l'une complètement écrite en Java (équipement C) et l'autre implémentée de façon native (équipement E). Comme figuré dans le tableau 6-6, la version purement logicielle d'AES est plutôt lente (environ 0,4 s par bloc chiffré), surtout lorsqu'il est nécessaire d'initialiser une clé (1,3 s). Les ressources cryptographiques additionnelles (fonctions EAX et OMAC.) sont fournies par les classes Java.

On constate également à travers le même tableau que le temps de réalisation de la fonction OMAC croit proportionnellement avec le nombre de blocs AES. La pénalité Java par bloc (notée **p** dans le tableau 6-6) est d'environ 20 ms dans le cas de l'équipement E, et d'environ 50 ms pour l'équipement C.

tableau 6-6 : paramètres de base pour EAP-PSK.

	Temps de déduction des 2 clés AES (s)	Chiffrement AES T_{AES} (s)	OMAC 25 octets 3 x AES $T=D_{AES}(T_{AES}+p)$ (s)	OMAC 49 octets 5 x AES $T=D_{AES}(T_{AES}+p)$ (s)	EAX (N=16 octets, H=5 octets, M=1 octet) 10 x AES (s)
E	0,018	0,011	0,114	0,170	1,022
C	2,6	0,39	1,30	2,24	5,13

Le tableau 6-7 donne des détails sur les temps observés. La procédure EAX est plus complexe et induit davantage de temps additionnel de calcul d'environ 900 ms pour l'équipement E, et de 1230 ms pour l'équipement C. On note que le temps total de traitement de l'équipement E (2,45 s) est principalement dû à l'*overhead* Java parce que la pénalité AES (34 x 0,011 = 0,374 s) n'est pas le facteur prépondérant.

tableau 6-7 : temps mesurés lors d'une session d'authentification EAP-PSK.

		Equipement E (AES réalisé en natif)		Equipement C (AES réalisé en Java)	
Opération	**Nombre de blocs chiffrés AES (n)**	**Temps Calcul CT (s)**	**Estimation "overhead" Java CT-n.T_{AES} (s)**	**Temps Calcul CT(s)**	**Estimation "overhead" Java CT-n.T_{AES} (s)**
1re requête					
SET_KEY(AK)		0,018		2,60	
OMAC (49 octets)	5	0,170	0,110	2,24	0,29
2e requête					
OMAC (25 octets)	3	0,114	0,079	1,30	0,13
SET_KEY(KDK)		0,018		2,60	
MCM	6	0,070		2,34	
SET_KEY(TEK)		0,018		2,60	
EAX(N=16, H=5, M=1)	10	1,022	0,906	5,13	1,23
EAX(N=16, H=5, M=1)	10	1,022	0,906	5,13	1,23
TOTAL	**34**	**2,452**	**2,001**	**23,94**	**2,88**

IV.4. Résultats avec la méthode EAP-AKA

Dans [UrDa06] nous employons une approche plus formelle pour évaluer les performances de la carte employant EAP-AKA, approche qui par ailleurs a été utilisée pour confirmer les résultats de [UrDa05] concernant les méthodes EAP-TLS. La seule carte disposant en natif de l'algorithme de AES (équipement dénommé E) est ici employée. Nous avons convenu d'appeler :

- $T_{EAP\text{-}AKA}$ le temps total associé à la réalisation de la méthode EAP-AKA ;
- $T_{Transfer}$ le temps nécessaire au transport des paquets EAP entre le terminal qui contrôle la carte à puce et l'application qui s'exécute dans la carte à puce ;
- T_{Crypto} le temps total consommé à la réalisation des opérations cryptographiques ;
- $T_{SoftwareOverhead}$ le temps utilisé par toutes les autres opérations ;
- T_{AES} le temps moyen de chiffrement d'un bloc au moyen de la fonction AES ;
- T_{Digest} le temps moyen de calcul de l'empreinte chiffrée d'un bloc.

Les différents coûts relatifs à l'ouverture réussie d'une session EAP-AKA peuvent alors être exprimés au moyen des formules suivantes :

(1) $T_{EAP\text{-}AKA} = T_{Transfer} + T_{Crypto} + T_{SoftwareOverhead}$

(2) $T_{Crypto} = 5xT_{AES} + 18xT_{Digest}$

Mais en tenant compte du fait que lors de l'implémentation logicielle de la fonction PRF, on s'est basé sur 5 valeurs modifiées de SHA1, on peut préciser l'expression du temps consommé par les ressources cryptographiques comme suit :

(3) $T'_{Crypto} = 5xT_{AES} + 13xT_{Digest} + T_{PRF}$

tableau 6-8 : performances de EAP-AKA pour la carte Java E.

EAP-AKA Time $T_{EAP-AKA}$ (s)	Tranfer 108 octets Time $T_{Transfer}$ (s)	$5xT_{AES}$ f1…f5 Time (s)	$13xT_{Digest}$ HMACs and XKEY (s)	PRF Time T_{PRF} (s)	Overhead Time $T_{SoftwareOverhead}$ (s)
5,95	<0,2	0,056	0,064	5,65	>0

En considérant T_{Digest} = 4,8 ms et T_{AES} = 11,3 ms, les éléments de performances du protocole EAP-AK ont été calculés pour la carte Java E. Comme il apparaît dans le tableau 6-8, la majeure partie du temps est consommée par la fonction PRF. Le protocole EAP-AKA pourrait donc être très efficace si la fonction PRF était disponible au sein des ressources API. Et dans ces conditions, il n'est pas impossible que le temps de traitement d'une authentification par le protocole EAP-AKA soit inférieur à une demi-seconde.

V. Conclusion

La plate-forme *OpenEapSmartcard* est réalisable ; nous en avons donné la preuve grâce à son implémentation dans une demie douzaine de cartes à puce Java du marché provenant de fabricants différents d'une part, et supportant différentes solutions d'authentification basées sur le protocole EAP d'autre part. La conception modulaire de la plate-forme a facilité cette adaptation. Les résultats obtenus démontrent que, même avec des cartes à puce qui ne sont pas conçues au départ pour, la plate-forme peut y être déployée. De toute évidence, la performance est fonction des constituants des cartes et de la complexité des algorithmes qu'elles sont amenées à héberger. On gagnerait très certainement à voir implémentées de façon native dans les cartes des fonctions comme par exemple celle calculant les nombres pseudo-aléatoires de différentes tailles ou celle réalisant le chiffrement RC4.

Parmi les cartes Java classiques testées, on en trouve dont les temps RTT sont inférieurs à 30 secondes et peuvent par conséquent être employées pour l'authentification dans des infrastructures sans-fil IP existantes. Les premiers résultats semblent indiquer que l'authentification par EAP-PSK est plus rapide que celle avec EAP-TLS *resume mode*. Cependant globalement, les délais d'authentification sont encore très importants en comparaison avec les solutions logicielles classiques ; une situation qui serait très probablement en rapport d'abord avec l'absence de certaines API Java, et ensuite avec les performances limitées des composants (capacités de la mémoire RAM et de traitement). Comme cette architecture fonctionne avec des cartes Java standard et qu'il est très probable que leurs performances suivent la loi de Moore, la carte à puce peut devenir une alternative crédible vis-à-vis des solutions logicielles traditionnelles. C'est pourquoi dores et déjà, nous entreprenons d'utiliser ce nouvel environnement pour héberger un serveur : c'est l'objet du prochain chapitre.

Chapitre 7 : Les micro-serveurs d'authentification

Pour répondre aux besoins de sécurité des données personnelles de l'utilisateur dans les environnements de mobilité sans fil IP, l'une des solutions consiste à notre avis à utiliser la carte à puce comme lieu de stockage et aussi de traitement, ce que nous avons présenté dans le chapitre concernant EAP et les cartes à puce. Puisque l'accès à ce lieu doit être bien réglementé, nous estimons qu'il est impérieux d'y mettre un serveur EAP dont la seule raison d'être serait de permettre l'ouverture d'un canal sécurisé entre le monde extérieur à la carte à puce, et la carte à puce elle-même ; c'est l'objet de ce chapitre qui débute avec un point sur ce qui motive la création des micro-serveurs d'authentification. Nous donnons ensuite une description de ce que pourrait être l'architecture de ces micro-serveurs. La troisième partie du chapitre présente une implémentation de ces micro-serveurs d'authentification accompagnée des résultats des tests de performances récemment obtenus. Le chapitre se termine sur une évocation des perspectives qu'ouvre ce nouveau concept de micro-serveur d'authentification qui a fait l'objet d'une publication [UrDaBa05].

I. Motivations

La sécurité d'accès aux réseaux sans fil émergents et bientôt présents dans tous nos lieux de vie constitue un défi qu'il faut rapidement relever. Autant les organisations offrant les infrastructures sans fil IP doivent contrôler et gérer les accès à leurs réseaux, ne serait-ce que pour des raisons légales (à cause de l'interdiction de certains comportements dans les réseaux sans fil IP ouverts), autant les utilisateurs accédant à ces infrastructures doivent avoir la garantie de ne pas exposer involontairement au monde leurs données intimes, leurs habitudes, leurs comportements et fréquentations aussi bien des lieux réels (cybercafés, hot spots, etc.) que virtuels (sites Internet). Sur un ordinateur, fut-il de poche, ces risques de fuites d'informations sont grands. Par exemple, certains des obstacles actuels au déploiement des architectures centralisées de PKI pourraient être contournés avec ces micro-serveurs qui ouvriraient la voie à une décentralisation sécurisée (distribution et conservation des certificats et des clés privées, mise à jour des listes de révocation).

Dans la perspective d'une multiplication tous azimuts des usages des réseaux sans fil, les contextes/profils de l'utilisateur vont aussi se diversifier. De nombreux documents précieux pour l'utilisateur devront être disponibles et accessibles de façon instantanée et presque transparente lorsque celui-ci accède aux réseaux sans fil IP. Pour gérer cet ensemble de documents confidentiels de l'utilisateur dans un environnement inviolable tel que celui de la carte à puce, il est bon de disposer sur la carte elle-même d'un serveur capable de rendre pareil service.

Un autre besoin à l'origine d'un serveur d'authentification correspond à celui d'être en mesure de générer en toute confiance des attestations ou reçus de connexion comportant des informations utiles comme l'identité d'un client et l'estampille horodatée de sa session sur le réseau sans fil IP. L'idée de base est que dans des environnements pervasifs, les accès aux réseaux, même lorsqu'ils sont gratuits, doivent être enregistrés pour une question de respect vis-à-vis de la loi. Nous appelons service de traces le service chargé d'enregistrer les informations sur la session grâce à l'utilisation de deux dispositifs infalsifiables, l'un du côté du client et l'autre du côté du serveur. Pour qu'un serveur de

traces puisse bien répondre à ces objectifs, il lui faut disposer d'un canal sécurisé de communication.

Les micro-serveurs d'authentification ont pour vocation d'utiliser le contexte de la carte à puce, ce dispositif inviolable que l'utilisateur transporte très aisément dans son portefeuille ou dans son téléphone portable, pour rendre de nouveaux services. Du fait de leur transportabilité et de leur résistance aux falsifications, ces équipements peuvent servir au stockage d'informations personnelles à l'utilisateur (identités, certificats, profiles, mots de passe, pseudonymes, etc.) ; des informations qui pourront lui servir utilement dans son nomadisme et sa mobilité.

Les micro-serveurs d'authentification constituent le premier exemple d'hébergement d'un serveur dans une carte à puce. Le service attendu de ces micro-serveurs est celui de l'ouverture d'un canal de communication sécurisé avec d'autres entités. Ce service représente à nos yeux la première brique à mettre en place si l'on désire continuer de faire de la carte à puce ce coffre-fort électronique mobile. De nombreux protocoles comme EAP-TLS, EAP-SIM, EAP-AKA, EAP-PSK sont adaptés à l'ouverture d'un canal sécurisé (avec des données chiffrées et signées), pourvu que l'environnement où ils pourront être utilisés soit créé : c'est l'objectif des micro-serveurs d'authentification EAP.

II. Description générale

Les micro-serveurs d'authentification sont d'abord des serveurs, en ce sens qu'ils sont destinés à rendre des services ; et les services à rendre sont ceux de l'authentification mutuelle. Ils sont micro parce que résidant dans des cartes à puce. Pour leur réalisation on s'est appuyé sur les acquis en matière de cartes à puce EAP, de cartes à puce EAP-TLS et de plate-forme *OpenEapSmartcard*.

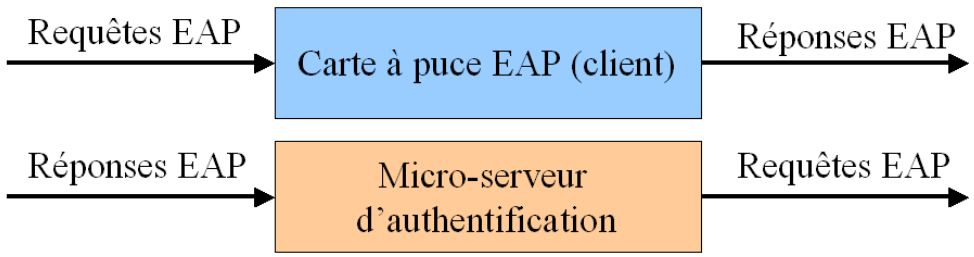


figure 7-1 : fonctionnement des cartes à puce client EAP et micro-serveur EAP.

En effet, les cartes à puce EAP réalisées depuis 2003 ont permis de démontrer que le contexte particulier de la carte à puce pouvait servir à la sécurisation des accès aux réseaux. La première carte à puce a hébergé un client EAP dont le fonctionnement consistait à recevoir des requêtes provenant d'un serveur EAP présent sur un terminal, de les analyser et de produire en retour les réponses correspondantes. En fait, ces cartes, à travers les requêtes EAP, ont été interrogées selon le protocole EAP par rapport à ce qu'elles savaient : clés, certificats, numéros de session, etc. ; et les réponses engendrées n'étaient que la restitution d'informations qu'elles possédaient déjà (voir figure 7-1). Ces cartes généralement placées dans des situations de passivité ne "prennent pas l'initiative" d'agir par elles-mêmes. Elles sont sollicitées et ce n'est qu'après cela qu'elles réagissent ; d'ailleurs on parle souvent de cartes réactives.

Les cartes à puce que nous présentons comme micro-serveurs d'authentification, sont différentes en ce sens qu'elles hébergent plutôt un serveur EAP. On rappelle que couramment, le contexte des serveurs d'authentification AAA est celui dans lequel se retrouvent les serveurs EAP. Le rôle du serveur EAP est de produire des messages de requêtes EAP, puis d'analyser les messages de réponse qu'il reçoit en retour ; ce sont par conséquent des entités duales des clients EAP. Elles sont actives puisque ce sont elles qui prennent l'initiative de "poser les questions EAP".

figure 7-2 : micro-serveur et client EAP insérés chacun dans un lecteur USB de cartes à puce.

La représentation d'une plate-forme expérimentale, constituée de deux cartes à puce Java, de deux lecteurs de cartes à puce USB et d'un composant logiciel que nous appelons pont EAP (*EAP-Bridge*) est donnée à la figure 7-2. C'est elle qui servira à faire la preuve que le concept de micro-serveur n'est pas que théorique (*proof of concept*). Précisons que le pont permet principalement de relayer les messages entre la carte à puce "serveur EAP" et la carte à puce "client EAP" et de visualiser le déroulement des échanges.

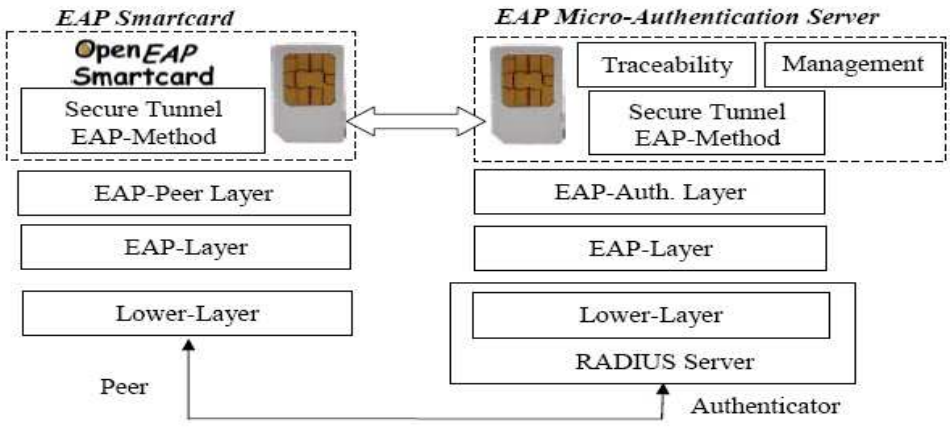


figure 7-3 : environnement logiciel d'une carte à puce EAP et d'un micro-serveur EAP.

A la figure 7-3 nous avons voulu, seulement à titre de comparaison, mettre en vis-à-vis les couches logicielles constituant un client et un serveur EAP. Comme déjà expliqué, le client EAP est d'habitude sur la machine qui adresse le point d'accès dans le réseaux sans fil IP. Le serveur EAP est quant à lui sur le serveur d'authentification AAA qui, lui, partage un lien sécurisé généralement par RADIUS avec le point d'accès. Comme on peut s'en apercevoir sur cette figure, les architectures logicielles du client EAP et du serveur

EAP sont légèrement différentes (*EAP-Peer Layer # EAP-Auth Layer*), mais elles ont des fonctionnements symétriques. Ainsi, le moteur *EapEngine* dans le contexte du serveur EAP reçoit des messages de réponse *EAP-Response* qu'il achemine vers la méthode appropriée, ce que ne fait pas *EapEngine* dans le client EAP. De plus, si dans le cas de la carte à puce EAP les méthodes traitent les messages *EAP-Request* que le moteur *EapEngine* leur confie, les méthodes dans le cas du serveur EAP ont pour mission de traiter les messages *EAP-Response* délivrés par le moteur *EapEngine*.

L'emploi de la plate-forme *OpenEapSmartcard* a beaucoup facilité la réalisation de ces micro-serveurs d'authentification, du fait encore une fois de leur grande adaptabilité. Les descriptions faites au chapitre 4 sur le protocole EAP et la carte à puce restent valables en ce qui concerne les serveurs EAP même lorsqu'ils sont abrités dans des cartes à puce.

III. Tests de performances

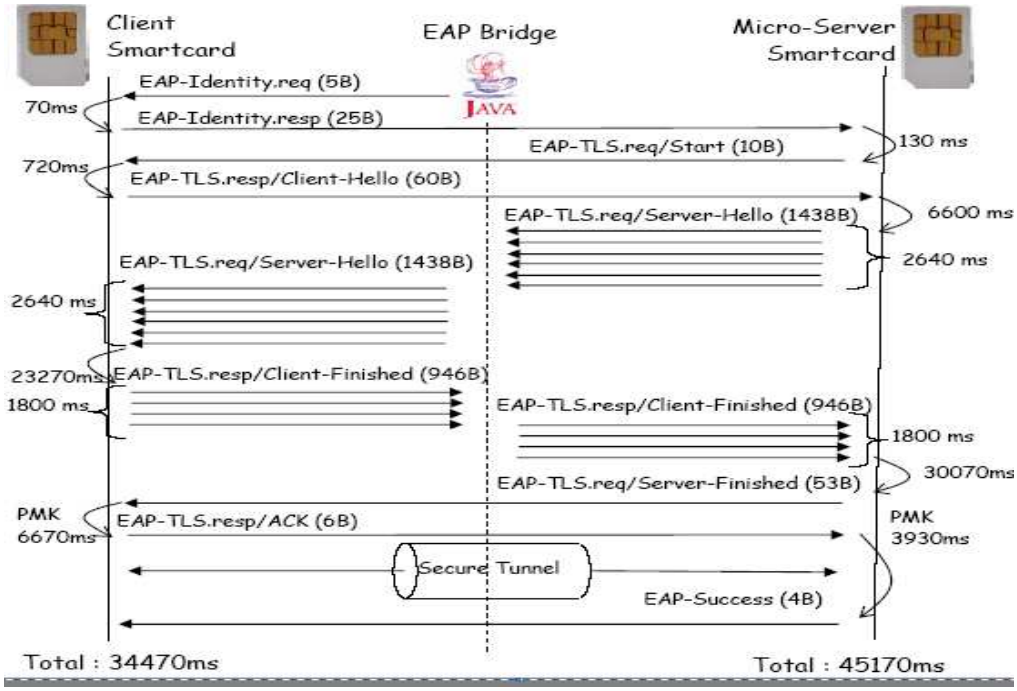


figure 7-4 : authentification entre un client EAP Jcop et un micro-serveur EAP Jcop.

La figure 7-4 ci-dessus présente une session d'authentification EAP entre un client et un micro-serveur. Dans cet environnement, un constituant Java additionnel (Pont EAP sur la figure 7-4) est inséré entre les deux équipements infalsifiables. Ce pont est à l'origine du démarrage de l'échange, et procure aussi les fonctionnalités additionnelles comme déjà décrites lors du commentaire de la figure 7-2. Pour les deux dispositifs qui sont identiques, les temps de traitement sont assez voisins, bien que les opérations au niveau du serveur soient 30% plus lentes ; ceci est principalement causé par le message *Server-Hello* qui demande plus de ressources (comme en particulier des opérations d'écriture dans la mémoire non volatile) qu'un message *Client-Hello*. Dans cet exemple, un tunnel sécurisé est installé à la fin du processus d'authentification réussie. Et ce mécanisme permet d'échanger des messages en toute sécurité entre les deux entités d'extrémité, ce qui pourrait ouvrir la voie à des opérations de gestion et de traçabilité.

Du point de vue logiciel, une application serveur d'authentification EAP est très voisine de celle d'un client EAP. La charge cryptographique est sensiblement la même, mais le traitement des messages est significativement différent.

tableau 7-1 : comparaison des performances du client et du serveur EAP-TLS.

	Cartes employées	
	C (Jcop)	D (Gemplus)
$T_{EAP\text{-}TLS}$ Client (s)	33,8	10,4
$T_{EAP\text{-}TLS}$ Server (s)	45,2	13,0
% différence de performance relative	25,22	20

A travers le tableau 7-1, nous présentons les mesures de performances pour les cartes à puce C et D employées alternativement comme clients et serveurs. On remarque que les serveurs EAP-TLS usent d'un temps additionnel tendant vers 30%. Nous attribuons pour l'instant cette différence à des opérations de lecture et d'écriture au niveau de la mémoire non volatile d'informations extra mais nécessaires lors de la construction ou de la concaténation des données employées en entrée des fonctions de hachage par exemple. Ces explications devront être prochainement vérifiées.

IV. Perspectives

Puisque les ondes radio ne peuvent pas être confinées dans un espace hermétiquement clos, il est nécessaire de créer des mécanismes de sécurité plus adaptés comme :

- L'instauration d'un contrôle des accès au réseau qui exige l'authentification des visiteurs. En fonction de leurs lettres de crédits, ils bénéficieront des services auxquels ils ont souscrit.
- La signature des paquets ; un visiteur authentifié échange des trames qui sont identifiées par l'adresse MAC et l'adresse IP. La signature des messages protège contre les attaques par usurpation d'une part, et assure la non répudiation obligatoire dans un service de facturation d'autre part.
- La confidentialité de l'information ; les données du visiteur sont chiffrées afin d'en garantir la protection contre les écoutes indiscrètes et les renifleurs. Dans certains cas, le chiffrement est réalisé au niveau de la couche application.

C'est dire que de nombreux champs d'exploitation des résultats obtenus existent, qu'il s'agisse du monde des réseaux filaires ou qu'il s'agisse du domaine du sans fil.

IV.1. Serveurs de traces ou d'attestations de connexion

Les micro-serveurs introduisent de nouveaux paradigmes utiles dans la sécurité des réseaux sans fil IP. Etant donné que les scénarii d'authentification se trouvent réalisés sur des plate-formes de confiance, il devient possible d'envisager une architecture décentralisée basée sur des paradigmes de PKI (en employant EAP-TLS par exemple). Les

micro-serveurs et les cartes à puce EAP peuvent alors partager la même autorité de certification, et de ce fait se faire confiance mutuellement. L'un des intérêts essentiels des dispositifs infalsifiables est de conserver des données personnelles sensibles dans des environnements protégés, des informations qui pourraient inclure les attestations de connexion (utiles pour les fonctionnalités de traçabilité) ou des clés cryptographiques variées.

Nous nommons traçabilité la capacité pour aussi bien le client que le serveur, de générer un document de connexion qui prouve une authentification réussie. C'est ce document de connexion que nous nommons attestation ou reçu de connexion. Sa constitution pourrait inclure les paramètres suivants :

- les identités du client et du serveur, en considérant par exemple leurs certificats X.509 ;
- des données en rapport avec la connexion : une date au format UNIX, le condensé de tous les messages côté client chiffré avec la clé de session et signé (message *Client-finished*), le condensé de tous les messages côté serveur chiffré avec la clé de session et signé (message *Server-finished*) ;
- les signatures du client et du serveur.

Ces informations de l'attestation de connexion sont échangées via un canal sécurisé et conservées dans des cartes à puce EAP et des micro-serveurs d'authentification. En utilisant une clé RSA courante de 1024 bits, la taille d'un tel document avoisine 3 ko. Il est donc prévisible que les capacités de stockage des cartes à puce elles-mêmes soient insuffisantes pour le stockage d'un grand nombre de ces attestations de connexion. Pour cela, nous envisageons que ces justificatifs de connexion puissent être encapsulés dans des *blobs* (blocs binaires) sécurisés dont les clés cryptographiques seront protégées par les clés privées du client ou du serveur selon les mécanismes introduits par les TPM [TPM04]. Un *blob* contiendrait donc les éléments suivants :

- une attestation de connexion et son condensé, le tout chiffré par une clé symétrique, $\{\text{attestation}, \textit{digest}(\text{attestation})\}_{BlobKey}$;
- la clé *BlobKey* chiffrée au moyen de la clé RSA privée, $\{BlobKey\}_{PrivateKey}$.

IV.2. Micro-serveurs pour WLAN et VPN

Selon le standard 802.11i, un serveur d'authentification ne devrait jamais exposer à une tierce entité la clé symétrique (le secret RADIUS par exemple) qu'il partage avec le point d'accès par lequel passe un demandeur d'accès donné, ce qui signifie que ce serveur ne doit jamais être compromis. Qu'il soit physiquement sur la même machine que le point d'accès ou bien que le lien physique reliant le serveur et le point d'accès appartiennent au même domaine administratif que lui, cet environnement doit être hautement sécurisé. Cette affirmation découle du fait que si un point d'accès et un serveur d'authentification ne sont pas physiquement sur la même machine ou bien ne partagent par directement à eux-deux la clé KEK (*Key Encryption Key*) de chiffrement des clés, il est impossible de garantir au demandeur d'accès que la clé fournie par le serveur au point d'accès n'a jamais été compromise. C'est donc un lieu d'utilisation idéal des micro-serveurs d'authentification.

Ces composants pourraient être introduits dans une variété de classes de serveurs, comme les serveurs d'accès employés lors de l'authentification par PPP, les serveurs IKE [IKEv2_05] permettant d'échanger des clés pour créer des tunnels sur Internet (VPN) et faciliter le déploiement du protocole IPsec [IPSEC98], les serveurs RADIUS

(authentification dans une architecture WiFi), serveurs EAP-PSK devant conserver les paires (*Session-Id*, *Master-secret*) dans un environnement sécurisé et protégé de sorte à contrer les attaques consistant à récupérer le *Master secret*. Ces serveurs améliorent tous la sécurité grâce au protocole EAP dans un contexte de traitement informatique hautement sécurisé conservant précieusement les matériaux cryptographiques. Même dans le cas où le système d'exploitation du serveur est compromis, les données critiques comme les clés privées RSA résistent à l'espionnage.

IV.3. Distributeur de clés dans un WPAN ou WLAN

L'idée est de permettre à des équipements appartenant à un même réseau, de pouvoir communiquer simplement, en récupérant auprès du distributeur de clés, la clé qu'il faut pour pouvoir communiquer avec une entité donnée. Ce n'est bien évidemment envisageable que pour un réseau de taille réduite. Son fonctionnement ressemblerait à un serveur DHCP mais plutôt pour les clés d'authentification.

IV.4. Carte à puce EAP bi-mode

Il s'agit d'une carte à puce offrant à la fois des services de client EAP que de serveur EAP. Dans ce cas les attributs d'identité servent à initier une session particulière. Par la suite cette session pourra être référencée par un index comme cela a été introduit dans [OpnEap05]. Une partie des données de cette carte ne serait utilisable que par le client EAP (demandeur de services), tandis que l'autre partie ne le serait que pour le serveur EAP (fournisseur de services).

IV.5. Carte à puce SIM biométrique

Elle repose sur l'idée que l'empreinte digitale d'un individu puisse être stockée dans sa carte SIM. En fait, on imagine qu'au moment de la signature du contrat avec l'opérateur de téléphonie mobile, on demande à l'abonné d'apposer son doigt sur un dispositif de saisie de son empreinte qui ne sera stockée nulle part ailleurs que sur sa puce. Et l'authentification nécessaire à la connexion à des réseaux se fera en utilisant un dispositif capable à partir d'une photographie du doigt prise par son téléphone portable de comparer avec ce qui est stocké sur la puce. Ce type d'authentification peut être un service offert lors de l'accès aux réseaux WiFi et WiMax, pourvu que les fournisseurs de ces accès passent un contrat avec les opérateurs de téléphonie mobile.

IV.6. Serveurs EAP-SIM mobiles

Il y a la possibilité de développer des serveurs mobiles qui combinent les facilités du GSM et celles des serveurs EAP, étant donné que la plate-forme *OpenEapSmartcard* fonctionne sur les cartes SIM. On pourrait de cette manière, par interrogation, récupérer à distance (sous réserve d'accord avec l'opérateur – par exemple le service CPT "C'est Pour Toi" au Burkina Fasopar l'un des opérateurs de téléphonie mobile permettant de transférer une partie de son crédit d'appel vers un autre abonné) des données hébergées sur une autre carte SIM (client EAP-SIM) présente dans la poche d'un abonné. Ce même serveur pourrait de façon autonome, récupérer des informations disponibles dans la zone

géographique où il se retrouve, et cela auprès de clients EAP qu'il peut interroger (point d'accès d'un hot spot, BTS d'un réseau GSM, etc.)

IV.7. Coffre-fort triplement blindé

L'idée ici c'est que tout ce qui est stocké sur la carte le soit de façon chiffrée. Pour cela, on a sur la carte un module qui ne peut être interrogé qu'à la suite d'une authentification mutuelle réussie. Et sur la base de la clé PMK (ancienne ou nouvelle), un document peut être conservé chiffré sur la carte. Au moment de sa récupération la fourniture de la PMK et aussi du condensé en rapport avec le document recherché est obligatoire. Le triple blindage résulte d'abord de l'emploi de l'espace de la carte à puce, ensuite de l'exploitation d'un canal sécurisé pour y avoir accès, et enfin de l'utilisation d'une clé dérivée de la PMK pour protéger la confidentialité et l'intégrité du document.

V. Conclusion

Dans ce chapitre, nous avons introduit le premier micro-serveur bâti sur la plate-forme *OpenEapSmartcard*. Ce nouvel environnement de confiance que nous proposons permet d'assurer la gestion des cartes à puce EAP et d'offrir de nombreux services comme par exemple ceux attachés à la génération des attestations de connexion, à la distribution des clés en vue de la création de tunnels sécurisés.

Le suivi du fonctionnement de tous ces serveurs devra pouvoir être assuré à distance et de manière non rébarbative ; c'est l'objet du chapitre suivant portant sur l'introduction des services Web au sein des cartes à puce *OpenEapSmartcard*.

Chapitre 8 : Le TEAPM, une nouvelle architecture pour l'administration OTA dans les réseaux sans fil IP

N'importe quel serveur, si l'on souhaite qu'il continue à rendre correctement le service qui est le sien, doit pouvoir être administré régulièrement. Or, comme ces micro-serveurs ont été réalisés sur des cartes à puce, il faut pouvoir d'une part en assurer l'administration courante à distance, mais aussi travailler à rendre plus commodes toutes les opérations dans cet environnement, qu'elles soient purement administratives ou qu'elles concernent l'accueil de nouveaux services. C'est l'objet de la proposition que nous faisons avec le module TEAPM (*Trusted EAP Module*), un dispositif logiciel au cœur duquel se loge le protocole EAP, et ayant fait l'objet de la publication [UrDa06].

Dans ce chapitre, nous commençons par présenter le TEAPM aussi bien en termes d'architecture de protocoles que de services qu'il est en mesure de rendre. Nous montrons ensuite une utilisation de ce dispositif pour réaliser des serveurs EAP dans des cartes à puce. Nous donnons les possibilités d'administration et les facilités de déploiement de nouveaux services que laisse entrevoir le module TEAPM. Enfin, le chapitre s'achève avec la présentation des résultats d'une expérimentation employant un serveur RADIUS couplé à une batterie de serveurs EAP utilisant le TEAPM.

I. Architecture protocolaire du TEAPM

L'architecture TEAPM (*Trusted EAP Module*) fait suite aux résultats obtenus premièrement en termes de performances des cartes à puce, et deuxièmement en termes d'ouverture de l'environnement de la carte à puce avec *OpenEapSmartcard*. Elle s'inscrit aussi dans la perspective des évolutions futures dans le domaine des technologies de la carte à puce relativement à la loi de Moore, pour transformer les cartes à puce en de véritables coffres-forts de poche électroniques administrables à distance.

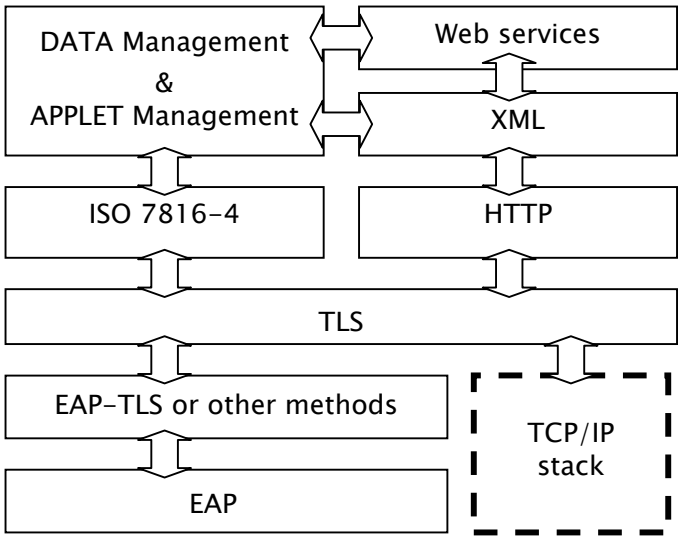


figure 8-1 : pile protocolaire du TEAPM.

Le TEAPM est un dispositif logiciel infalsifiable au cœur duquel on a le protocole EAP et ses méthodes d'authentification robustes (voir figure 8-1) fonctionnant dans un environnement de traitement informatique fiable. Quoique dans notre travail on ne fasse usage que de la carte à puce comme cadre de son déploiement, nous pensons que d'autres implémentations physiques du TEAPM sont envisageables.

La présence dans cette pile protocolaire de EAP et de la méthode EAP-TLS (ou toute autre méthode d'authentification robuste comparable) assure la réalisation d'une authentification mutuelle dont l'heureux aboutissement est la mise en place d'un canal d'échanges sécurisés. Le déroulement de cette opération dans un espace inviolable rend possible l'acheminement et le stockage sécurisés de documents électroniques précieux tels que des clés, des certificats, des numéros de compte, des mots de passe, des profils, etc. au sein de la carte à puce *OpenEapSmartcard*. De cette manière, nous offrons aux utilisateurs un composant électronique de poche qui fonctionnellement, ressemble aux indéboulonnables TPM (*Trusted Platform Module*) développés par le TCG (*Trusted Computing Group*) dans le cadre de la sécurisation des plate-formes de traitements informatiques que sont les ordinateurs [TPM05].

Un démultiplexeur au-dessus de la couche EAP-TLS permet d'envoyer les messages TLS provenant de la couche EAP vers une entité capable soit de produire directement des APDU, soit travaillant avec le protocole HTTP (*Hyper Text Transfer Protocol*).

La présence de la couche ISO 7816-4 d'un côté en dessous de la couche application (figure 8-1) permet de maintenir l'ouverture de la plate-forme en la gardant compatible avec les applications utilisant directement des APDU. Signalons que cette couche est ainsi intitulée parce que nous avons employé comme environnement d'hébergement infalsifiable la carte à puce ; nous aurions utilisé un autre environnement garantissant cette propriété que cette couche aurait été désignée autrement. L'essentiel, c'est que cette couche assure le fonctionnement de base de l'équipement de traitement et de stockage inviolable employé.

La présence des protocoles HTTP 1.1 [HTTP1.1_99] et XML (*eXtended Markup Language*) [XML1.0_04] de l'autre côté de la couche application donne une allure novatrice à notre pile protocolaire. En effet, la plupart des travaux de recherche concernant l'emploi de la carte à puce dans le domaine des réseaux informatiques sont orientés vers une utilisation de ce dispositif en tant que nœud Internet (cf. chapitre 4), en y incluant une pile TCP/IP [STEV94] ; mais les capacités de traitement et de stockage de l'équipement ont jusque là limité ce type d'utilisation. Dans le TEAPM, nous court-circuitons TCP/IP pour faire entrer les nouveaux services Web au sein de la carte à puce, aussi bien en termes de clients que de serveurs.

HTTP est un protocole de la couche application OSI employé sur les systèmes d'information multimédia distribués et collaboratifs, fonctionnant selon le modèle requête/réponse. Sa première utilisation remonte à 1990 avec l'initiative du *World Wide Web global information*. HTTP suppose de disposer d'un transport fiable ; et il peut reposer sur tout protocole fournissant cette garantie. Sa version 1.1 est la première qui soit fiable et prenne en compte la présence des proxys hiérarchiques, des caches, le besoin de connexions persistantes ou de machines hôtes (serveur) virtuelles, la découverte par deux applications qui communiquent de leurs capacités respectives, l'emploi d'une même connexion pour un ou plusieurs échanges de requêtes/réponses quoiqu'une connexion puisse être fermée pour des raisons variées. Le format des messages HTTP 1.1 est semblable à celui employé pour transmettre des documents de format quelconque

multimédia au sein d'un système de messagerie [MIME96] qui à son tour s'appuie sur le format d'échange des messages sur le premier Internet [ARPA82].

XML est le format universel de description des informations. Il est un sous-ensemble du langage SGML (*Standard Generalized Markup Language*) permettant d'employer des balises et des attributs pour formater des données. A la différence du langage HTML (*HyperText Markup Language*) réservé à la mise en forme d'un texte et dont les balises sont prédéfinies, en XML aucune restriction n'existe à employer des balises et des attributs pour marquer un document. L'une des caractéristiques les plus utiles du langage XML est son codage au format textuel, mais rien n'empêche d'échanger des contenus compressés ou chiffrés.

Avec l'architecture TEAPM, l'authentification a lieu avant une quelconque attribution d'adresse Internet, ce qui n'est pas le cas couramment. Il est donc impossible de bénéficier d'une quelconque ressource dans pareille architecture, tant que la phase d'authentification n'est pas correctement franchie.

Dans l'architecture TEAPM, la couche transport est celle reposant sur EAP/EAP-TLS. Grâce à elle il est possible d'offrir un service comparable à celui assuré par la présence de SMS-PP au niveau du GSM. En effet, sa présence dans une carte à puce employée au niveau de la sécurité d'accès d'un réseau sans fil IP offre la possibilité d'en faire un module d'identification/authentification utilisable et administrable à distance et objet d'une administration OTA.

L'emploi de la pile EAP/EAP-TLS peut dans un contexte d'échanges point à point, venir en remplacement de la pile protocolaire TCP/IP. Dans le même contexte, la pile EAP/EAP-TLS/TLS/HTTP peut être utilisée en remplacement de HTTPS. Et du coup, un accès aux ressources par l'intermédiaire d'une interface Web sécurisée est de ce fait possible.

La présence des standards HTTP et XML renforce aussi bien cette caractéristique d'ouverture souhaitée depuis l'initiative *OpenEapSmartcard*, que cette quête d'interopérabilité qui est une condition au succès dans un monde d'informatique communicante. C'est, nous le pensons, une des voies pour un développement et une utilisation des technologies Web pour administrer la carte à puce.

II. Services du TEAPM

Le module TEAPM est conçu comme un environnement tenant compte des plus récentes évolutions en matière d'authentification par le protocole EAP. Les services TEAPM vont être définis ici comme ceux fournis par des applications exécutées au sein d'une carte à puce Java offrant un environnement hautement sécurisé de traitement informatique. En plus de ceux qui pourront y être rajoutés par des utilisateurs, nous pensons qu'au moins doivent y être prévues les deux catégories suivantes de services : celle en rapport avec les aspects réseau d'une part, et celle liée à l'exploitation même du module d'autre part.

Dans les services réseaux se retrouvent les méthodes à l'interface entre les mécanismes d'authentification EAP et la couche *Peer-layer*. Ils sont décrits dans le [RFC4137] et concernent essentiellement l'initialisation des méthodes d'authentification donnée, l'acheminement et le traitement des paquets EAP par la méthode d'authentification à laquelle il sont destinés.

En plus des services réseau, ceux en rapport avec l'exploitation du module sont exigés pour le déploiement dans la pratique du TEAPM ; ce sont :

- La gestion des contenus ; c'est l'ensemble des facilités dont on a besoin pour télécharger les lettres de crédits utilisées par une méthode d'authentification donnée (certificats X.509, clés cryptographiques, etc.).
- La gestion de la sécurité ; ce service gère les mécanismes qui restreignent l'emploi du TEAPM aux seules utilisateurs autorisés (usage du code PIN, ou de données biométriques, etc.).
- La gestion des identités ; ce service permet, dans le cas où plusieurs méthodes sont disponibles, de n'en choisir qu'une.

Ces aspects sont davantage détaillés dans [EAPSup06].

III. Exemple de déploiement du TEAPM

La figure 8-2 présente un contexte WiFi réel fonctionnant avec un client et un serveur TEAPM. Nous décrivons succinctement les opérations qui ont lieu durant l'authentification du client :

1. Un ordinateur personnel PC client qui tente d'accéder aux ressources d'un réseau sans fil, transmet de façon périodique une trame EAP-Start afin de démarrer un scénario d'authentification.
2. Le point d'accès lui envoie ensuite une requête EAP d'identité qui est reçue par le système d'exploitation du PC qui à son tour la transmet au client EAP sur la carte à puce. Ce dispositif en retour fournit une identité au travers d'une réponse *EAP-Response/identity* qui est renvoyée jusqu'au point d'accès par l'intermédiaire du système d'exploitation du PC.

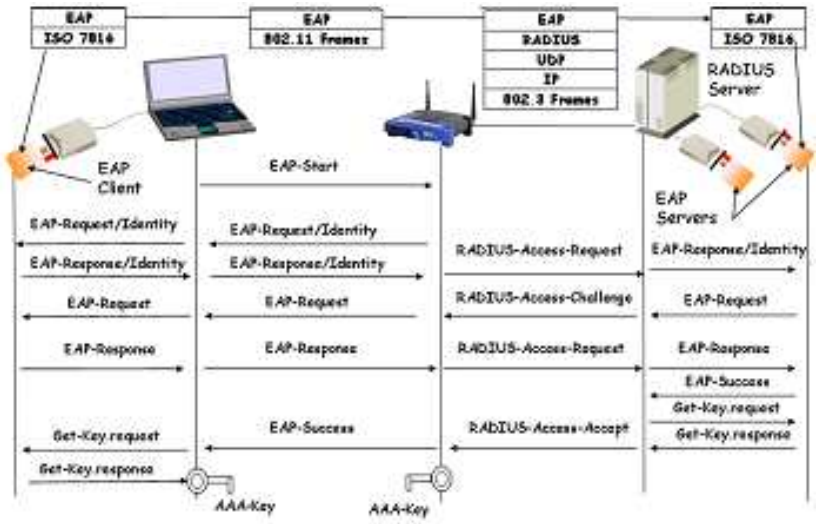


figure 8-2 : exemple de déploiement du TEAPM.

3. Cette réponse reçue par le point d'accès est encapsulée dans un paquet RADIUS [RFC3579] qu'il achemine vers le serveur RADIUS. Le logiciel RADIUS vérifie la disponibilité d'un serveur EAP sur carte à puce, auquel il délivre dans le cas échéant la réponse d'identité EAP : c'est le début d'une nouvelle session.
4. Le serveur EAP renvoie un message de requête EAP encapsulé dans un paquet RADIUS au point d'accès, qui ensuite le transmet au PC, lequel finalement remet le message au client EAP de la carte à puce. Celui-ci traite la requête et produit en retour une réponse qui est expédiée vers le serveur EAP sur la carte à puce.

Le dialogue d'authentification qui s'établit est un ensemble de questions/réponses échangées entre un client EAP et un serveur EAP. A la fin de cet échange, le serveur EAP produit un message *EAP-Success*, et comme résultat le serveur RADIUS obtient une clé cryptographique principale MSK, appelée aussi clé AAA, requise pour la sécurité de toutes les opérations entre le point d'accès et le client PC.

La clé MSK est de manière sécurisée descendue vers le point d'accès, en même temps que le message *EAP-Success* qui est encore acheminé vers le client.

A la réception de ce message, le client récupère la clé MSK de sa carte à puce EAP. Tout est alors bon pour calculer tous les éléments cryptographiques employés par les protocoles de sécurité sur la radio tels que WEP [802.11_99] ou 802.11i [802.11i_04].

IV. Administration et déploiement de nouveaux services

Comme déjà indiqué, la gestion des cartes à puce est une activité classique dans les réseaux GSM aujourd'hui, ce qui facilite l'administration des données et des applications. Dans le contexte du TEAPM, les fonctions de gestion exécutées après une authentification mutuelle réussie (figure 8-3) vont rendre les services suivants :

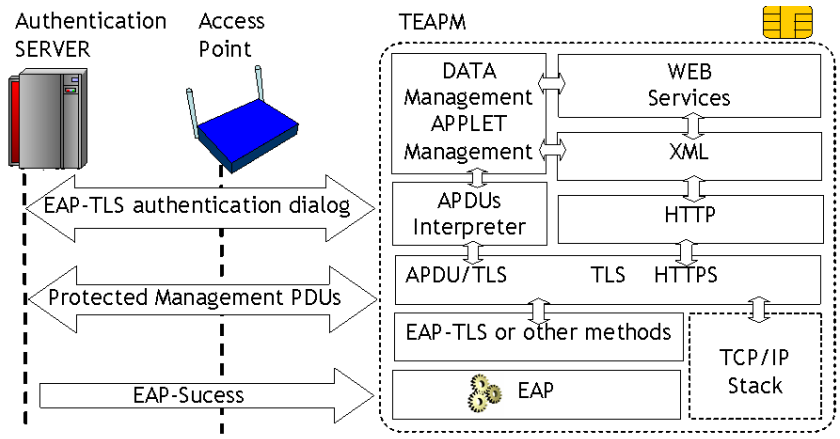


figure 8-3 : environnement de gestion à distance du TEAPM.

- L'invalidation ou la révocation des lettres de crédits comme les certificats X.509 et les clés privées associées. En raison de l'extrême difficulté à dupliquer les cartes à puce, seul un exemplaire physique de ces entités existe. La possibilité de bloquer leur emploi à distance est une exigence importante de sécurité dans un environnement distribué de PKI.
- La mise à jour des lettres de crédits permettant de garantir une continuité ou une extension ou une réduction des services souscrits par les abonnés. Cela se fait en remplaçant ou en rajoutant des éléments d'information contrôlant le fonctionnement de ces services.
- Le téléchargement de nouvelles applications du fait que les protocoles d'authentification peuvent avoir besoin de faire évoluer et d'inclure de nouvelles fonctionnalités. Dans ce genre de situation, le logiciel est de manière transparente mis à jour, sans que par exemple les porteurs de TEAPM n'aient à intervenir.

Ces fonctionnalités intéressantes nécessitent un transfert de données sous protection entre le TEAPM et le serveur administrant le TEAPM. La plupart des méthodes d'authentification établissent des liaisons sécurisées transportant des messages chiffrés et signés. Ainsi, la couche *Record layer* du protocole EAP-TLS [TLS99] peut envoyer des AVP (*Attribute Value Pair*) chiffrés, et EAP-AKA [EAPAKA06] peut transporter une charge utile chiffrée. Quoique la présence du protocole DHCP induise des contraintes de temps (habituellement 60 s) au cours de la première authentification, le processus de ré-authentification comme cela est défini dans l'architecture IEEE 802.1X n'est pas gêné par ce problème. D'ailleurs c'est un moyen pratique pouvant servir à mettre à jour les informations contenues dans le dispositif infalsifiable.

La syntaxe employée pour gérer couramment les cartes à puce est celle fondée sur les APDU. Puisque toutes les interfaces fonctionnelles du TEAPM sont décrites avec un ensemble d'APDU, l'administration du TEAPM présenté marche avec un interpréteur d'APDU (voir figure 8-4) qui analyse et exécute les commandes APDU en entrée. Cependant, d'autres techniques peuvent également convenir et éviter le déploiement d'outils de gestion particuliers ; nous proposons dans ce cadre deux candidats potentiels :

- LDAP (*Ligthweight Directory Access Protocol*) [LDAP97]. Le contenu du TEAPM peut être considéré comme une entrée d'un annuaire portable. Les entités de gestion interviennent comme des clients LDAP qui créent, mettent à jour ou suppriment des attributs en utilisant des messages codés selon la syntaxe ASN.1 et transportés dans des paquets EAP-TLS ;
- Les services Web. La nouvelle génération de cartes à puce [TuCoSo05] offre assez de ressources en termes de capacité de traitement et de stockage, de sorte à pouvoir supporter une pile TCP/IP et le protocole HTTP. De tels équipements peuvent renfermer des serveurs/clients Web et des facilités additionnelles XML, rendant les services embarqués dans la carte à puce administrables à partir des interfaces de services Web. Une première forme de ces services pourrait être réalisée par une DTD (*Document Type Definition*) spécifique traduisant un ensemble d'APDU en un document XML. C'est alors la porte ouverte à l'emploi des langages de description de services comme WSDL (*Web Services Description Langage*) [WDSL01] basés sur XML, comme SOAP (*Simple Object Access Protocol*) [SOAP1.1_00] le mécanisme d'échange d'informations structurées et typées entre des entités distribuées sur le Web défini par le W3C (*Web Consortium*).

L'usage des TEAPM au niveau des systèmes qui les accueillent, qu'ils soient clients ou serveurs, permet en cas de compromission du système d'exploitation, de conserver la confidentialité et l'intégrité des données critiques (clés privées RSA par exemple) qu'ils gardent. La possibilité de leur administration à distance peut servir à leur désactivation. Plus encore, lorsque l'identité du client est cachée par des champs cryptographiques, ces composants peuvent vérifier secrètement ce paramètre et notifier le succès de l'authentification sans divulguer la véritable identité à l'utilisateur du réseau. Cette propriété facilite l'introduction des PET (*Privacy Enhancing Technologies*) dans le déploiement des réseaux pervasifs émergents.

Contrairement aux opérateurs de téléphonie mobile gérant des réseaux mondiaux, le développement des infrastructures IP sans fil force à penser qu'on aura affaire plutôt à des constellations de petits domaines gérés par des autorités publiques (villes, campus, etc.), des compagnies privées ou même des individus. C'est très probable que chacune d'elles contrôlera les accès à son réseau en fonction de mécanismes et polices particuliers.

Le TEAPM qui est destiné à un large déploiement sans surcoût dû à la centralisation ou à un changement d'équipement, devrait très certainement aider à cette administration décentralisée.

V. Performances expérimentales

Dans une infrastructure IEEE 802.1X, les messages EAP sont transportés entre d'une part un point d'accès (AP) désigné dans la terminologie RADIUS par serveur d'accès au réseau (NAS), et d'autre part un serveur d'authentification (AS) via le protocole RADIUS. La sécurité de ce transport est basée sur un secret dit s*ecret RADIUS* partagé entre l'AP et le serveur RADIUS. De façon classique, il est difficile d'évaluer la sécurité physique d'un AP ; le secret est en permanence conservé sur l'AP, peut-être parfois sans aucune protection cryptographique. Pour cette raison nous avons aussi choisi de concevoir un serveur RADIUS, s'exécutant sur un ordinateur personnel (ce qui veut dire que la sécurité RADIUS est sous le contrôle de l'ordinateur personnel qui contient aussi le secret RADIUS), mais déléguant toutes les opérations EAP aux serveurs EAP (voir figure 8-4). Chacun de ces serveurs EAP exécute la méthode EAP-TLS associée à un unique certificat X.509 et sa clé privée RSA. Ces choix simplifient beaucoup la conception du logiciel.

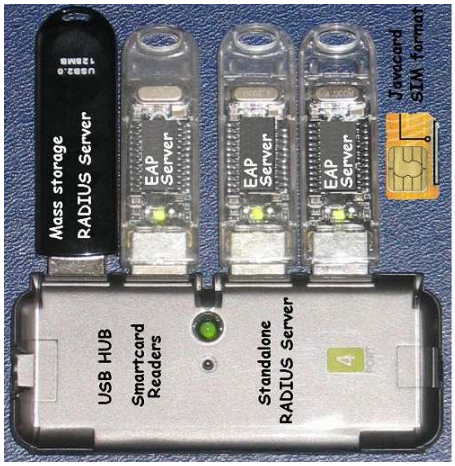


figure 8-4 : mise en œuvre pratique de serveurs TEAPM.

Chaque session EAP est identifiée de façon unique par un identificateur de session nommé *Session-Id* obtenu par la concaténation de deux valeurs, l'identificateur du NAS dénommé *NAS-Identifier* (Attribut numéro 32 selon [RFC2865]) et l'identificateur de la machine demandant l'accès au réseau désigné par *Calling-Station-Id* (l'adresse MAC du client correspondant à l'attribut numéro 31 dans [RFC2865]) de la manière suivante :

Session-Id = NAS-Identifier | | Calling-Station-Id.

Lorsqu'une nouvelle session démarre (réception d'un paquet RADIUS contenant une indication *EAP-Identity.response*), le logiciel du serveur RADIUS essaie de trouver un serveur EAP disponible auquel il confiera le déroulement de la session jusqu'à la fin. Si aucun serveur EAP n'est trouvé disponible, le paquet RADIUS est purement ignoré ; autrement tout le reste du processus d'authentification se poursuivra avec ce serveur EAP.

Une batterie de TEAPM peut alors être employée, et ainsi permettre le déroulement

de sessions EAP en parallèle, un peu à la manière de ce qui a cours dans des environnements de grille de processeurs. A la figure 8-5 nous proposons une illustration de ce qu'on pourrait nommer une grille de quatre TEAPM où à un même moment, chaque TEAPM pourrait être soit en train d'analyser un paquet de réponse EAP qui lui arrive, soit en train de réaliser en interne les traitements cryptographiques avec les matériaux en son sein en vue de produire un paquet de requête EAP.

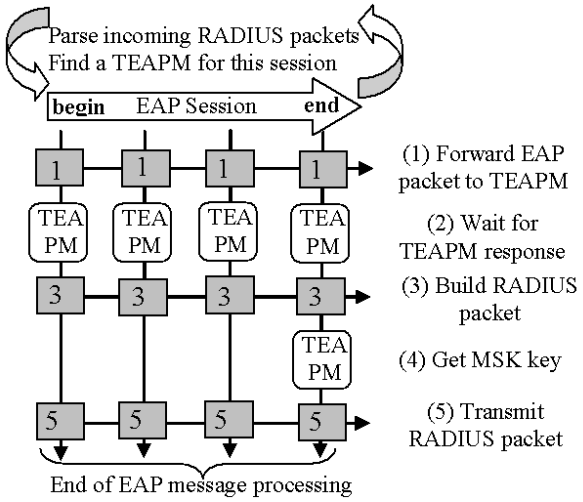


figure 8-5 : vue logique des opérations dans une grille de TEAPM.

Le traitement d'un message EAP est plutôt lent car nécessite quelques secondes, tandis qu'une session complète EAP coûte environ 5 s au TEAPM. De ce fait, chaque paquet EAP est géré par un fil (*thread*) logiciel qui attend la réponse du TEAPM, une réponse qui est par la suite encapsulée dans un paquet RADIUS envoyé au bon point d'accès.

Le fait qu'un paquet RADIUS soit ignoré en cas d'indisponibilité d'un serveur EAP ressemble à la situation habituelle de blocage survenant dans les centraux téléphoniques où un appel donné est ignoré dès lors qu'aucune ligne en sortie n'est disponible. En faisant l'hypothèse que le système fonctionne selon un modèle de file M/M/c/c, la formule d'Erlang-B s'écrit :

$$p_c = \frac{(\lambda/\mu)^c}{c!}\left[\sum_{k=0}^{c}\frac{(\lambda/\mu)^k}{k!}\right]^{-1}$$

avec comme :

p_c la probabilité de blocage (celle qu'un paquet RADIUS soit ignorée),

c le nombre de serveurs EAP sur cartes à puce,

λ le taux de sessions d'authentification, et

$1/\mu$ le temps moyen que dure une session d'authentification.

Avec notre meilleur couple de dispositifs (client et serveur), nous mesurons une durée d'authentification d'environ 5 + 5 = 10 s ; par conséquent $1/\mu = 10$. En faisant l'hypothèse que le réseau a 1000 utilisateurs, avec une authentification toutes les heures, on peut déduire que $\lambda = 1000/3600$, et que $\lambda/\mu = 10 \times 1000 / 3600 = 2{,}8$.

La probabilité de blocage p_c est de 50 % avec deux cartes à puce EAP (c=2), et seulement de 1 % avec 8 cartes à puce (c=8).

VI. Conclusion

Nous avons dans ce chapitre présenté une nouvelle architecture, celle du module TEAPM, qui dans notre expérimentation est logé dans une puce infalsifiable capable de réaliser en son sein le protocole EAP. Son interface fonctionnelle compatible avec les spécifications en cours d'élaboration à l'IETF, et l'ouverture de la plate-forme permettent de concevoir des composants de faible coût, aussi bien côté client que côté serveur, et dotés de capacités d'administration à distance. A titre expérimental, une implémentation d'un serveur RADIUS muni d'un serveur EAP construit avec le TEAPM a été réalisée. Les performances auxquelles on aboutit montrent qu'à l'heure actuelle, avec même les cartes à puce du marché, on peut réaliser des protocoles complexes comme EAP-TLS en moins de 5 s. Par conséquent, un déploiement des TEAPM est possible dans les réseaux actuels.

L'ouverture du TEAPM permet d'envisager son utilisation d'une part dans des cartes à puce SIM, donnant de ce fait l'opportunité d'une double administration *Over The Air*, et d'autre part dans ce grand chantier mondial actuellement en cours de mise en place de services Web distribués.

CONCLUSION GENERALE – PERSPECTIVES

Le choix de retenir la carte SIM pour d'une part contenir et produire des informations permettant d'assurer la sécurité d'accès aux services du réseau GSM, et d'autre part assurer une administration *Over The Air*, est à notre avis l'une des raisons de la rapidité avec laquelle le GSM a été largement adopté de par le monde par les utilisateurs. Ce réseau de radiotéléphonie de grande envergure est celui des opérateurs ; il fonctionne relativement bien avec des solutions de sécurité propriétaires.

Avec les constellations de réseaux sans fil IP apparaissant dans nos villes, fruits de l'interconnexion de réseaux sans fil IP de taille évoluant des WPAN (*Wireless Personal Area Network*) à l'envergure des WMAN *(Wireless Metropolitan Area Network*) en passant par les classiques WLAN (*Wireless Local Area Network*), la couverture géographique du réseau Internet est en train de ressembler à celle du réseau GSM. Cependant, une grande différence existe : ce réseau n'est pas celui d'un opérateur, mais sous la responsabilité d'une multitude d'administrateurs locaux plus ou moins indépendants. Or, qui dit diversité d'autorités administratives dit aussi diversité de solutions pour sécuriser l'accès à chacun d'entre eux, avec malgré tout, ces impératifs en termes d'interrelations, d'interopérabilité, etc.

Les spécifications sur la sécurité des réseaux sans fil IP préconisent l'utilisation de nombreuses et diverses clés pour l'authentification, le chiffrement et la garantie de l'intégrité des échanges. Mais en aucun moment, ces spécifications ne proposent l'emploi d'un module pour les calculer et les conserver dans un contexte sécurisé. Nous pensons qu'avec l'accroissement des performances des cartes à puce, il faut envisager de les utiliser pour réaliser au sein des réseaux sans fil IP la même fonction qui leur est assignée au niveau des réseaux GSM, à savoir un module d'identification et d'authentification.

L'implémentation du protocole EAP au sein de la carte à puce a inauguré le processus d'introduction de la carte à puce comme dispositif de sécurisation de l'accès aux réseaux sans fil IP. Nous avons voulu continuer le processus en proposant une méthode d'authentification simple mais robuste, ressemblant à un TLS allégé, capable de servir à la

fois dans un contexte cryptographique symétrique et asymétrique. C'était malheureusement sans compter avec la loi de Moore qui a donné les capacités nécessaires à l'implémentation au sein de la carte à puce Java du très réputé protocole EAP-TLS.

Dès lors, nos efforts se sont portés sur la définition d'une plate-forme en mesure de supporter des mécanismes d'authentification divers et variés qui, facilement, pourraient être déployés dans de nombreux contextes, et avec une plus grande ré-utilisabilité. Cela s'est traduit par le projet de la plate-forme ouverte dénommée *OpenEapSmartcard* proposant une architecture logicielle supportable par les cartes Java du marché et facilement réutilisable. Au sein de cette plate-forme se retrouvent quatre composants : le moteur EAP chargé de réaliser les méthodes d'authentification, la classe des méthodes définissant les scénarii spécifiques d'authentification, l'interface d'authentification définissant tous les services dont la présence est obligatoire pour que l'entité chargée des méthodes EAP puisse bien coopérer avec le moteur EAP, et enfin les classes de lettres de crédits qui vont chacune être associées à une classe donnée de méthodes et renfermer toute l'information nécessaire à la réalisation d'un scénario d'authentification spécifique. Les premiers résultats de la réalisation de cette plate-forme révèlent d'une part que des problèmes d'interopérabilité liés à la diversité des cartes Java du marché existent, et d'autre part que les performances actuelles de certaines cartes à puce du marché permettent de réaliser un scénario d'authentification EAP-TLS dans des délais intéressants.

Si pour délivrer des matériels cryptographiques nécessaires à la réalisation des mécanismes et protocoles de sécurité s'appuyant sur la carte à puce, l'hébergement dans la carte à puce d'un client EAP en mesure de répondre aux requêtes reçues est indispensable, nous sommes allés plus loin en proposant que ces cartes puissent aussi héberger un serveur EAP. Agissant comme acteur au sein de la carte à puce, le serveur EAP est l'initiateur des requêtes d'informations contextuelles fiables auprès de clients EAP crédibles, requêtes émises de manière transparente à l'utilisateur. Ces informations conservées en lieu sûr peuvent servir à la production de preuves de toutes sortes. C'est par une adaptation de la plate-forme *OpenEapSmartcard* qu'a été réalisé le serveur EAP ; et du coup cette plate-forme a été enrichie avec la présence de l'interface de personnalisation. Un autre résultat, c'est l'augmentation d'environ 30 % du temps de traitement côté serveur EAP, pour une même méthode EAP, par rapport au client EAP.

Afin de faciliter l'administration à distance des cartes à puce dotées de la plate-forme *OpenEapSmartcard*, nous avons proposé un dispositif logiciel, le TEAPM (*Trusted EAP Module*), au cœur duquel se retrouvent les protocoles EAP et EAP-TLS. Surmonté d'une part d'un interpréteur d'APDU, et d'autre part des protocoles XML et HTTP, l'environnement de la carte à puce tout en restant compatible avec les applications écrites en langage de commandes ISO 7816-4, s'ouvre en direction du monde des applications Web distribuées. La richesse des environnements de développement d'applications Web peut de ce fait servir à la réalisation d'applications pour les cartes à puce.

Nous avons également montré des utilisations possibles de ce dispositif à travers une architecture possible de serveurs RADIUS. D'autres architectures (vrais serveurs RADIUS sur carte à puce, ou proxy RADIUS) sont à étudier dans le sens d'une amélioration des performances et des coûts. Mais à première vue, nous pensons que les serveurs EAP bâtis sur le TEAPM pourraient être introduits sur toutes les machines au niveau desquelles des informations sensibles sont conservées et calculées. N'y aurait-il pas une approche généraliste à adopter ?

D'autres travaux pourraient être consacrés à l'implémentation du TEAPM dans les cartes SIM/USIM (*UMTS Subcriber Identity Module*) . La jonction des transports sécurisés par SMS et EAP/TLS augmenterait les offres et la qualité des services à l'adresse de l'utilisateur, et la concurrence aidant, leurs coûts pour celui-ci tendrait à la baisse.

Si le TEAPM peut aider au déploiement de serveurs EAP dans l'objectif d'augmenter la sécurité de l'accès aux services sur les réseaux en général, il reste à définir, en conformité avec les législations des pays, ce qu'on peut légalement collecter, les preuves qu'on peut fournir, à quelle entité ces preuves peuvent être fournies, etc. De même, la batterie d'informations contenues dans une carte peut servir à la production d'un anonymat ; comment, quand, et pour qui les produire ? Telles pourraient être quelques autres pistes de recherche future.

BIBLIOGRAPHIE

[7816-1_98] ISO/IEC, "Information technology – Identification cards – Integrated circuit(s) cards with contacts – Part 1: Physical characteristics", ISO/IEC 7816-1:1998(E), First edition, 1998.

[7816-2_99] ISO/IEC, "Information technology – Identification cards – Integrated circuit(s) cards with contacts – Part 2: Dimensions and location of the contacts", ISO/IEC 7816-2:1999(E), First edition, 1999.

[7816-3_97] ISO/IEC, "Information technology – Identification cards – Integrated circuit(s) cards with contacts – Part 3: Electronic signals and transmission protocols", ISO/IEC 7816-3:1997(E), Second edition, 1997.

[7816-4_95] ISO/IEC, "Information technology – Identification cards – Integrated circuit(s) cards with contacts – Part 4: Interindustry commands for interchange", ISO/IEC 7816-4:1997(E), First edition, 1995.

[802.11_99] ISO/IEC, "Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", ISO/IEC 8802-11:1999(E), ANSI/IEEE std 802.11, 1999 edition, 1999.

[802.11a_99] IEEE, "Supplement to 802.11 (1999) High-Speed Physical Layer in the 5 GHz Band", IEEE Std 802.11a, 1999.

[802.11g_03] IEEE, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 4: Further Higher Data Rate Extension in 2,4 GHz Band", IEEE Std 802.11g-2003, 2003.

[802.11i_04] IEEE, "Supplement to Standard for Telecommunications and Information Exchange Between Systems – LAN/MAN specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Enhanced Security", IEEE standard 802.11i, 2004.

[802.16_04] IEEE, "IEEE Standard for Local and Metropolitan Area Networks. Part 16: Air Interface for Fixed Broadband Wireless Access Systems", IEEE Std 802.16-2004 (Revision of IEEE Std 802.16-2001), 2004.

[802.16e_05] IEEE, "Approved Draft IEEE Standard for Local and Metropolitan Area Networks. Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems. Amendment for Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands", IEEE 802.16e, December 2005.

[802.1X_01] Institute of Electrical and Electronics Engineers, "Local and Metropolitan Area Networks : Port-Based Network Access Control", IEEE Std 802.1X-2001, September 2001.

[AES01] National Institute of Standards and Technology, "Specification for the Advanced Encryption Standard (AES)", Federal Information Processing Standards (FIPS) 197, November 2001.

[ARPA82] D. Crocker, "Standard for The Format of ARPA Internet Text Messages", STD 11, IETF RFC 822, August 1982.

[ASN1_02] ITU-T | ISO/IEC, "Information technology – ASN.1 encoding rules - Specification of Basic Encoding Rules (BER), Canonical Encoding Rules

(CER) and Distinguished Encoding Rules (DER)", ITU-T Rec. X.690 (2002) | ISO/IEC 8825-1:2002, 2002.

[BaUr04] M. Badra, P. Urien, "Enhancing WLAN security by introducing EAP-TLS smartcards", proceedings of WWW/Internet, volume 1, pp 342-349, October 6-9, Madrid, Spain, 2004.

[BELPIC01] "Projet BELPIC (Belgian ELectronic Personal Identity Card)", 2001. http://www.belgium.be/eportal

[BoGoWa01] N. Borisov, I. Goldberg, D. Wagner, "Intercepting Mobile Communications: The insecurity of 802.11", Proceeding of the 11th Annual International Conference on Mobile Computing and Network, July 16-21, 2001.

[CARI04] M. Dandjinou, P. Urien, "Le protocole EAP-SSC", Actes du 7e Colloque Africain sur la Recherche en Informatique - CARI'04, 22-25 novembre, Hammamet, Tunisie, 2004.

[CCM03] D. Whiting, R. Housley, N. Ferguson, "Counter with CBC-MAC (CCM)", IETF RFC 3610, September 2003.

[CHEN00] Z. Chen, "Java Card Technology for Smart Cards: Architecture and Programmer's Guide", Sun book, 2000.

[COPS00] D. Durham, Ed., J. Boyle, R. Cohen, S. Herzog, R. Rajan, A. Sastry, "The COPS (Common Open Policy Service) Protocol", IETF RFC 2748, January 2000.

[DES80] National Institute of Standards and Technology, "DES modes of operation", FIPS PUB 81, December 1980.

[DES99] National Institute of Standards and Technology, "Data Encryption Standard (DES)", FIPS PUB 46-3, October 1999.

[DHCP97] R. Droms, "Dynamic Host Configuration Protocol", IETF RFC 2131, March 1997.

[DIAM03] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, J. Arkko, "Diameter Base Protocol", IETF RFC 3588, September 2003.

[DIFHEL03] T. Kivinen, M. Kojo, "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)", IETF RFC 3526, May 2003.

[draftSSC03] P. Urien, M. Dandjinou, "EAP-SSC Secured Smartcard Channel", draft-urien-eap-ssc-01.txt, December 2003. < Obsolete>.

[EAP04] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", IETF RFC 3748 (replaces the obsolete RFC 2284), June 2004.

[EAPAKA06] J. Arkko, H. Haverinen, "Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)", IETF RFC 4187, January 2006.

[EAPAX05] T. Clancy, W. Arbaugh, "EAP Password Authenticated Exchange", draft-clancy-eap-pax-05, <work in progress>, October 2005.

[EAPKey05] B. Aboba, "Extensible Authentication Protocol (EAP) Key Management Extensions", draft-aboba-eap-keying-extens-00.txt, April 2005.

[EAPKey06] B. Aboba, D. Simon, P. Eronen, H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP) Key Management Framework", draft-ietf-keying-09.txt, <work-in-progress>, January 2006.

[EAPPSK04] F. Bersani, "The EAP-PSK Protocol : a Pre-Shared Key EAP Method", IETF draft, draft-bersani-eap-psk-06, 2004.

[EAPSSC04] M. Dandjinou, P. Urien, "EAP-SSC Protocol", Proceedings of IEEE 3rd International Conference on Networking (ICN'2004), Gosier, Guadeloupe, French Carribbean, February 29 – March 4, ISBN: 0-86341-326-9, pp. 399-405, 2004.

[EAPSup06] P. Urien, G. Pujolle, "EAP-Support in Smartcard", draft-urien-eap-smartcard-10.txt, <work in progress>, February 2006.

[EAPTLS04] P. Urien, M. Badra, M. Dandjinou, "EAP-TLS Smartcards, from Dream to Reality", 4th Workshop on Applications and Services in Wireless Networks, ASWN'2004, Boston University, Boston, Massachusetts, USA, August 8-11, 2004.

[EAPTLS99] B. Aboba, D. Simon, "PPP EAP TLS Authentication Protocol", IETF RFC 2716, October 1999.

[FluMaSh01] S. Fluhrer, I. Mantin, A. Shamir, "Weakness in the Key scheduling algorithm of RC4", 8th Annual Workshop on Selected Areas in Cryptography, August 2001.

[GLOBPL03] Global Platform Card Specification version 2.1.1, March 2003

[GSM01.04] ETSI, "Digital cellular telecommunication system (Phase 2+); Abbreviations and acronyms (GSM 01.04 version 8.0.0 Release 1999)", ETSI TR 101 748, 2000.

[GSM02.02] ETSI, "Digital cellular telecommunications system (Phase 2+); Bearer Services (BS) supported by a GSM Public Land Mobile Network (PLMN) (GSM 02.02 version 7.0.2 Release 1998)", ETSI EN 300 904, 1999.

[GSM02.03] ETSI, "Digital cellular telecommunications system (Phase 2+); Teleservices supported by a GSM Public Land Mobile Network (PLMN) (GSM 02.03 version 7.0.0 Release 1998)", ETSI TS 100 905, 1999.

[GSM02.04] ETSI, "Digital cellular telecommunications system (Phase 2+); General on supplementary services (GSM 02.04 version 7.1.2 Release 1998)", ETSI EN 300 918, 1999.

[GSM02.07] ETSI, "Digital cellular telecommunications system (Phase 2+); Mobile Stations (MS) features (GSM 02.07 version 7.1.0 Release 1998)", ETSI TS 100 906, 2000.

[GSM02.09] ETSI, "Digital cellular telecommunications system (Phase 2+); Security aspects (GSM 02.09 version 8.0.1 Release 1999)", ETSI TS 100 920, 2001.

[GSM02.17] ETSI, "Digital cellular telecommunications system (Phase 2+); Subscriber Identity Modules (SIM); Functional characteristics (GSM 02.07 version 8.0.0 Release 1999)", ETSI TS 100 922, 2000.

[GSM02.48] ETSI, "Digital cellular telecommunications system (Phase 2+); Security mechanisms for the SIM Application Toolkit; Stage 1 (GSM 02.48 version 8.0.0 Release 1999)", ETSI TS 101 180, 2000.

[GSM02.90] ETSI, "Digital cellular telecommunications system (Phase 2+); Unstructured Supplementary Service Data (USSD) - Stage 1 (GSM 02.90 version 7.0.0 Release 1998", ETSI TS 100 625, 1999.

[GSM03.03] ETSI, "Digital cellular telecommunications system (Phase 2+); Numbering, addressing and identification (3GPP TS 03.03 version 7.8.0 Release 1998", ETSI TS 100 927, 2003.

[GSM03.19] ETSI, "Digital cellular telecommunications system (Phase 2+); GSM API for SIM toolkit stage 2 (3GPP TS 03.19 version 8.5.0 Release 1999", ETSI 101 476, 2002.

[GSM03.20] ETSI, "Digital cellular telecommunications system (Phase 2+); Security related network functions (GSM 03.20 version 8.1.0 Release 1999", ETSI TS 100 929, 2001.

[GSM03.38] ETSI, "Digital cellular telecommunications system (Phase 2+); Alphabets and language-specific information (GSM 03.38 version 7.2.0 Release 1998)", ETSI TS 100 900, 1999.

[GSM03.40] ETSI, "Digital cellular telecommunications system (Phase 2+); Technical realization of the Short Message Service (SMS) Point-to-Point (PP) (3GPP TS 03.40 version 7.5.0 Release 1998)", ETSI TS 100 901, 2001.

[GSM03.41] ETSI, "Digital cellular telecommunications system (Phase 2+); Technical realization of the Short Message Service Cell Broadcast (SMSCB) (3GPP TS 03.41 version 7.4.0 Release 1998)", ETSI TS 100 902, 2000.

[GSM03.42] ETSI, "Digital cellular telecommunications system (Phase 2+); Compression algorithm for text messaging services (GSM 03.42 version 7.1.1 Release 1998)", ETSI TS 101 032, 1999.

[GSM03.48] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Terminals; Security mechanisms for the SIM application toolkit; Stage 2 (Release 1999)", 3GPP TS 03.48 V8.9.0, 2005.

[GSM03.90] ETSI, "Digital cellular telecommunications system (Phase 2+); Unstructured Supplementary Service Data (USSD) - Stage 2 (GSM 03.90 version 7.0.0 Release 1998", ETSI 100 549, 1999.

[GSM04.10] ETSI, "Digital cellular telecommunications system (Phase 2+); Mobile Radio Interface Layer 3 - Supplementary Services specification; General aspects (3GPP TS 04.10 version 7.1.0 Release 1998", ETSI TS 100 941, 2001.

[GSM04.80] ETSI, "Digital cellular telecommunications system (Phase 2+); Mobile Radio Interface Layer 3 - Supplementary Services specification formats and coding (3GPP TS 04.80 version 7.4.1 Release 1998", ETSI TS 100 950, 2003.

[GSM11.11] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Terminals Specification of the Subscriber Identity Module – Mobile Equipment (SIM – ME) interface (Release 1999)", 3GPP TS 11.11 V8.13.0, 2005.

[GSM11.14] 3GPP, "3rd Generation Partnership Project; Specification of the SIM Application Toolkit for the Subscriber Identity Module – Mobile Equipment (SIM – ME) interface (Release 1999)", 3GPP TS 11.14 V8.17.0, 2004.

[GSM22.057] ETSI, "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); Mobile Execution Environment (MexE) service description; Stage 1 (3GPP TS 22.057 version 6.0.0 Release 6)", ETSI TS 122 057, 2005.

[GSM23.057] ETSI, "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); Mobile Execution Environment (MExE); Functional description; Stage 2 (3GPP TS 23.057 version 6.2.0 Release 6)", ETSI TS 123 057, 2003.

[GUYO_05] V. Guyot, "Smartcard, a mobility vector", Phd defense, September 30th 2005, University of Paris 6, Paris, France.

[HMAC97] H. Krawczyk, M. Bellare, R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", IETF RFC 2104, September 1997.

[HTTP1.1_99] R. Fielding, J Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hyper Text Transfer Protoco - HTTP/1.1", IETF RFC 2616, June 1999.

[IKEv2_05] C. Kaufman, Ed., "Internet Key Exchange (IKEv2) Protocol", IETF RFC 4306, December 2005.

[INES05] projet INES (Identité Nationale Electronique Sécurisée) de carte d'identité électronique. (en 2006 sur http://www.libertysecurity.org/article372.html)

[IPSEC98] S. Kent, R. Atkinson, "Security Architecture for the Internet Protocol", IETF RFC 2401, November 1998.

[LaGoTa00] X. Lagrange, P. Godlewski, S. Tabbane, "Réseaux GSM", 5e édition, Hermès Science Publications, Paris, 2000.

[LDAP97] M. Wahl, T. Howes, S. Kille, "Lighweight Directory Access Protocol (v3)", IETF RFC 2251, December 1999.

[MCM03] H. Gilbert, "The Security of One-Block-to-Many Modes of Operation", FSE 03, Springer-Verlag LNCS 2287, 2003.

[MD5_92] R. Rivest, "The MD5 Message-Digest Algorithm", IETF RFC 1321, April 1992.

[MILEN02] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Specification of the MILENAGE Algorithm Set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*; Document 2: Algorithm Specification", 3GPP TS 35.206 V5.0.0, June 2002.

[MIME96] K. Moore, "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", IETF RFC 2047, November 1996.

[NAI05] B. Aboba, M. Beadles, J. Arkko, P. Eronen, "The Network Access Identifier", IETF RFC 4282, December 2005.

[NIST01] FIPS PUB 197, "Advanced Encryption Standard (AES)", National Institute of Standards and Technology (NIST), November 2001.

[NIST97] FIPS PUB 186-2, " Digital Signature Standard (DSS)", National Institute of Standards and Technology (NIST), 1997.

[OCSP99] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP", IETF RFC 2560, June 1999.

[OMAC03] T. Iwata, K. Kurosawa, "OMAC: One-Key CBC MAC", FSE 03, Springer-Verlag LNCS 2887, 2003.

[OpnEap05] P. Urien, M. Dandjinou, "The OpenEapSmartcard project", short paper, Applied Cryptography and Network Security 2005, ANCS 2005, Columbia University, New York, USA, June 7-10, 2005.

[OpnEap06] Site Web OpenEapSmartcard , (en 2006 sur le site http://www.enst.fr/~urien/openeapsmartcard).

[PCSC96] "PC/SC, Interoperability Specification for Integrated ICCs and Personnel Computer Systems", © 1996 CP8 Transac, HP, Microsoft, Schlumberger, Siemens, Nixdorf, 1996.

[PIV05] NIST, "Personal Identity Verification of Federal Employees / Contractors", Federal Information Processing Standard (FIPS) 201, 2005. (en 2006 sur le site http://csrc.nist.gov/piv-program/)

[PKCS1_02] RSA Laboratories, "PKCS#1 v2.1: RSA Cryptography Standard", June 2002.

[PKCS3_93] RSA Laboratories, "PKCS#3: Diffie-Hellman Key Agreement standard", RSA Laboratories Technical Note Version 1.4 revised, November 1993.

[PKICRL99] R. Housley, W. Ford, W. Polk, D. Solo, "Internet X.509 Public Key Infrastructure Certificate and CRL profile", IETF RFC 2459, January 1999.

[PPTP99] K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, G. Zorn, "Point-to-Point Tunneling Protocol (PPTP)", IETF RFC 2637, July 1999.

[RESCO03] E. Rescorla, "SSL and TLS: Designing and Building Secure Systems", Addison-Wesley, ISBN:0-201-61598-3, 4th printing, August 2003.

[RFC1661] W. Simpson, "The Point-to-Point Protocol (PPP)", IETF RFC 1661, July 1994.

[RFC2865] C. Rigney, S. Willens, A. Rubens, W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", IETF RFC 2865, June 2000.

[RFC3579] B. Aboba, P. Calhoun, "RADIUS (Remote Authentication Dial In User Service Protocol) Support for Extensible Authentication Protocol (EAP)", IETF RFC 3579, September 2003.

[RFC4137] J. Vollbrecht, P. Eronen, N. Petroni, Y. Ohba "State Machines for Extensible Authentication Protocol (EAP) Peer and Authenticator", IETF RFC 4137, August 2005.

[RSA78] R. Rivest, A. Shamir, L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, 21 (2), pp. 120-126, February 1978.

[RSVP97] L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", IETF RFC 2205, September 1997.

[SaTh01] J. Sanchez, M. Thioune, "UMTS, services, architecture et WCDMA", Hermès Science Publications, Paris, 2001.

[SCH01] B. Schneier, "Secrets et mensonges – Sécurité numérique dans un monde en réseau", Vuibert, Paris, 2001.

[SHA1_02] National Institute of Standards and Technology, "Secure Hash Standard", Federal Information Processing Standards (FIPS) PUB 180-2, August 2002.

[SNMP02] D. Harrington, R. Presuhn, B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", IETF RFC 3411, December 2002.

[SOAP1.1_00] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, D. Winer, "Simple Object Access Protocol (SOAP) 1.1", W3C Note, May 2000. (en 2006 sous le lien http://www.w3.org/TR/2000/NOTE-SOAP-20000508)

[SSLv3_96] A. Freier, P. Karlton, P. Kocher, "The SSL protocol Version 3.0", Internet draft <draft-freier-ssl-version3-02.txt>, November 1996.

[STEV94] W. R. Stevens, "TCP/IP illustrated volume 1", Reading, Massachussetts, Addison-Wesley, 1994.

[SUN03] Sun Microsystems, "Java Card 2.2.1. API Specification", 2003. http://java.sun.com/products/javacard.

[SUN06] Sun Microsystems, "Java Card Plateform Specification 2.2.2", 2006. http://java.sun.com/products/javacard.

[TLS99] T. Dierks, C. Allen, "The TLS Protocol Version 1.0", IETF RFC 2246, January 1999.

[TLSPSK05] P. Eronen, H. Tschofenig, "Pre-Shared Key Ciphersuites or Transport Layer Security (TLS)", IETF RFC 4279, December 2005.

[TPM04] TCG, "TPM Specification version 1.2", Trusted Computing Group (TCG), 2004.

[TPM05] TCG, "TPM Main Part 1: Design Principles, Specification version 1.2 revision 85", 2005.

[TuCoSo05] J.P. Tual, A. Couchard, K. Sourgen, "USB Full Speed enabled smart cards for Consumer Electronics applications", Consumer Electronics, ISCE 2005, Proceedings of the 9th International Symposium, June 14-16, 2005, pp. 230-236.

[UR00] P. Urien, "Internet Card, a smart card as a true Internet node", Computer Communication, volume 23, issue 17, 2000.

[UrDa05] P. Urien, M. Dandjinou, "The OpenEapSmartcard platform", NETCON'05, Network Control and Engineering for QoS, Security and Mobility, IFIP TC6 Conference, Lannion, France, November 14-18, 2005.

[UrDa06] P. Urien, M. Dandjinou, "Designing Smartcards for Emerging Wireless Networks", CARDIS 2006, LNCS 3928, pp. 165-178, 2006.

[UrDaBa05] P. Urien, M. Dandjinou, M. Badra, "Introducing micro-authentication servers, in emerging pervasive environments", IADIS International Conference on WWW/Internet 2005, Lisbon, Portugal, ISBN : 972-8924-02-X, pp. 339-346, October 19-22, 2005.

[UrLo03] P. Urien, M. Loutrel, "The EAP smartcard. A tamper resistant device dedicated to 802.11 wireless networks", 3rd Workshop on applications and Services in Wireless Networks, Berne, Switzerland, July 2-4, 2003.

[UrTiLo02] P. Urien, A. Tizraoui, M. Loutrel, "Integrating EAP in SIM-IP smartcards", Second IEEE workshop on Applications and Services in Wireless networks, ASWN, Paris, Juillet 2002.

[WDSL01] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, "Web Services Description Langage (WSDL) 1.1", W3C Note, March 2001. (en 2006 sous le lien http://www.w3.org/TR/2001/NOTE-wsdl-20010315)

[XML1.0_04] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, François Yergeau, "Extensible Markup Language (XML) 1.0 (Third Edition)", W3C Recommendation, February 2004. (en 2006 sous le lien http://www.w3.org/TR/2004/REC-xml-20040204)

ANNEXE I

Hiérarchie des clés dans la spécification 802.16e

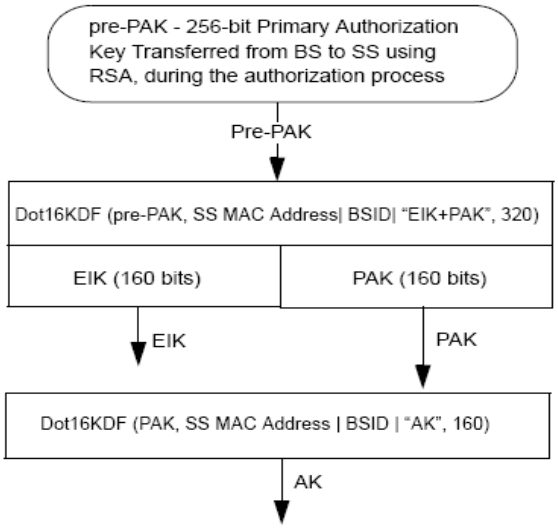


figure 8-6 : clé AK dérivée de PAK (phase d'autorisation basée sur RSA).

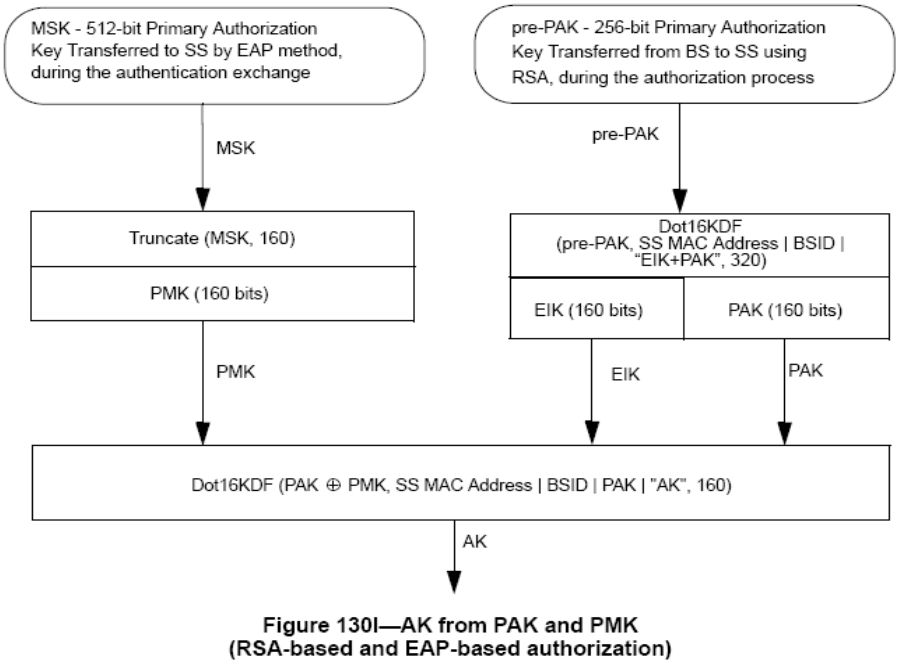


Figure 130l—AK from PAK and PMK (RSA-based and EAP-based authorization)

figure 8-7 : clé AK dérivée de PAK et PMK (autorisation basée sur RSA et EAP).

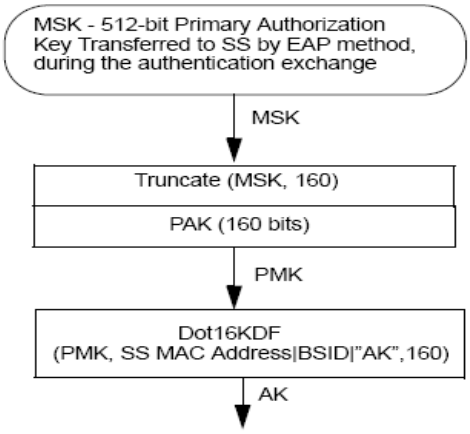


figure 8-8 : clé AK dérivée de PMK (autorisation basée sur EAP).

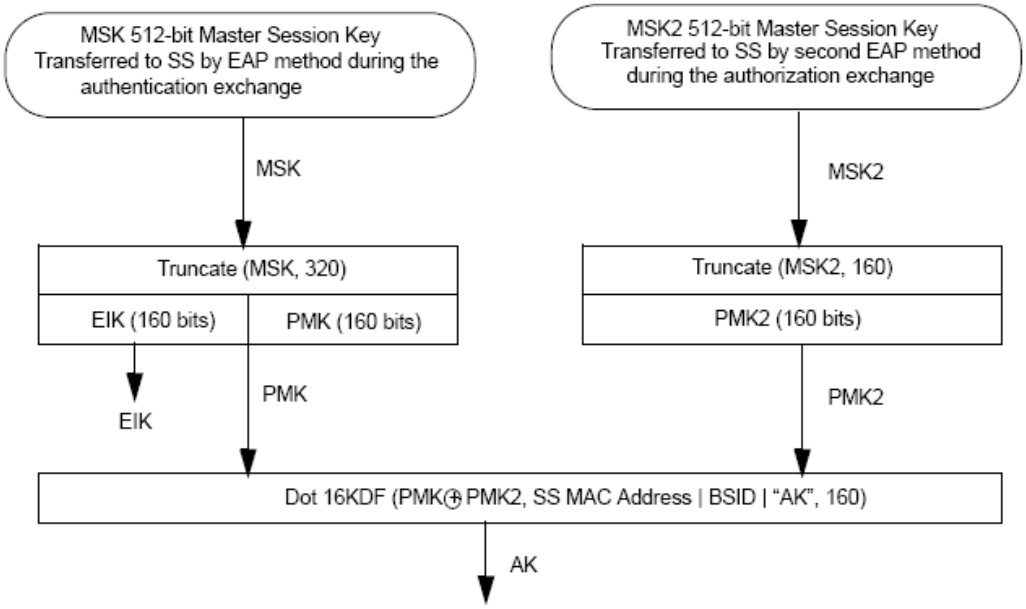


figure 8-9 : clé AK dérivée de PMK et PMK2 (autorisation et authentification basées sur EAP).

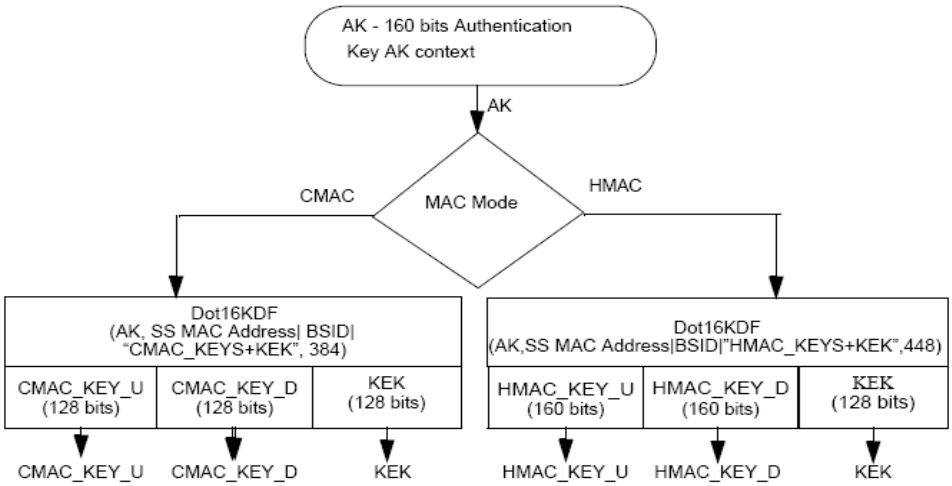


figure 8-10 : clés HMAC, CMAC et KEK dérivées de AK.

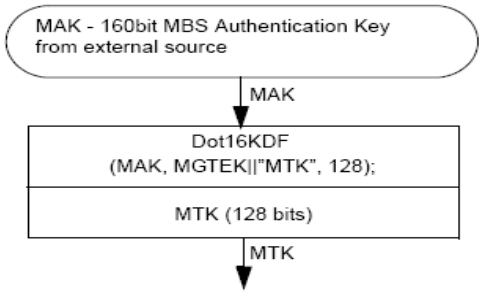


figure 8-11 : clé MTK dérivée de MAK.

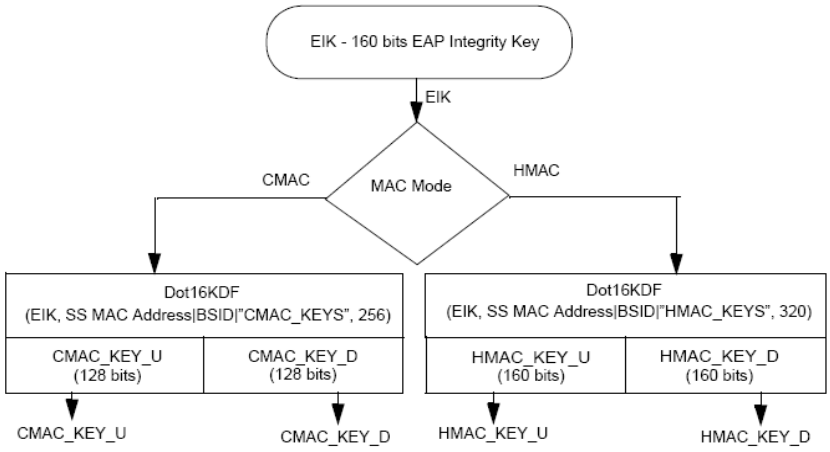


figure 8-12 : clés HMAC et CMAC dérivées de EIK.

ANNEXE II

Traces concernant les résultats de Simulation du protocole EAP-SSC

9 Examples of traces with plain text payload

This section of the document provides two examples of results of a simulation of the running of two sessions, the first session associated to a symmetrical key exchange context, and the second to a public key exchange context. All computed values used to produce the five packets which are exchanged in each case are presented with hereafter assumptions:
- EAP-SSC-Type equal 255 (hexadecimal FF);
- Starting Identifier equals 165 (hexadecimal A5);
- The payload is not enciphered.

9.1 Traces in a symmetrical key exchange context

```
//* value of the shared secret s
83D972D101F40973DEC8E32068B1DE581641EA76

//******** Beginning of the 1st packet of the exchange **********
//*********** sent by the Authentication Server (AS) ************
01 A5 00 1B FF 01 20                    ; header
 ^  ^ -----  ^  ^  ^
 |  |   ^    |  |  |
 |  |   |    |  |  +----- Flags field with S (Start) bit set
 |  |   |    |  +---- Sub-Type field set for symmetrical case
 |  |   |    +----- EAP-SSC-Type
 |  |   +------- Packet Length field set to 27
 |  +------- Identifier field
 +------- Code Field set for EAP-Request packet
BDD99CB2FDABDC5995521D3F4D7241BBA6A96E5D ; value of r1 (20 bytes)
//******************** End of the 1rst packet ********************

//* value of r2 (20 bytes) generated by the Smart Card
E72D5787D1C037E1DE3CFE63DCF5DF8DF2523693

//* value of D(r1 | s)
A575616DE4EB41230E39B28A94BB86039E27F8C9

//******** Beginning of the 2nd packet of the exchange **********
//************* sent by the Smart Card to the AS ***************
02 A5 00 1B FF 01 00                    ; header
 ^  ^ -----  ^  ^  ^
 |  |   ^    |  |  |
 |  |   |    |  |  +----- Flags field
 |  |   |    |  +---- Sub-Type field set for symmetrical case
 |  |   |    +----- EAP-SSC-Type
 |  |   +------- Packet Length field set to 27
 |  +------- Identifier field equals to that of the request packet
 +------- Code Field set for EAP-Response packet
425836EA352B76C2D0054CE9484E598E6C75CE5A ; Z = r2 XOR D(r1 | s)
//******************** End of the 2nd packet *********************

//* value of the computed Session's Key SK
AB5AFE7AC13CEE477BEACE3A5178AD9D7BD7D374

//******** Beginning of the 3rd packet of the exchange **********
//************* sent by the AS to the Smart Card ***************
01 A6 00 20 FF 01 08                    ; header
```

```
 ^  ^ -----  ^  ^  ^
 |  |   ^    |  |  |
 |  |   |    |  |  +----- Flags field with D (Digest) bit set
 |  |   |    |  +---- Sub-Type field set for symmetrical case
 |  |   |    +----- EAP-SSC-Type
 |  |   +------- Packet Length field set to 32
 |  +------- Identifier field has been incremented to 166
 +------- Code Field set for EAP-Request packet
68 65 6C 6C 6F ; M1="hello"
22F182938CBA24E4E49D2B5E9EA3B53321DE84FD ; D1 = D("hello" | SK)
//******************* End of the 3rd packet *********************

//******** Beginning of the 4th packet of the exchange **********
//************* sent by the Smart Card to the AS ****************
02 A6 00 20 FF 01 08                      ; header
 ^  ^ -----  ^  ^  ^
 |  |   ^    |  |  |
 |  |   |    |  |  +----- Flags field with D (Digest) bit set
 |  |   |    |  +---- Sub-Type field set for symmetrical case
 |  |   |    +----- EAP-SSC-Type
 |  |   +------- Packet Length field set to 32
 |  +------- Identifier field equals to that of the request packet
 +------- Code Field set for EAP-Response packet
77 6F 72 6C 64 ; M2="world"
AB10AB506D923CE0BC60221ACF503D6338C1EDA2 ; D2=D("world" | D1 | SK)
//******************* End of the 4th packet *********************

//******** Beginning of the 5th packet of the exchange **********
//************* sent by the AS (EAP-Success/End) ****************
03 A7 00 1F FF 01 18                      ; header
 ^  ^ -----  ^  ^  ^
 |  |   ^    |  |  |
 |  |   |    |  |  +----- Flags field with D (Digest) and E (End)
 |  |   |    |  |         bits set
 |  |   |    |  +---- Sub-Type field set for symmetrical case
 |  |   |    +----- EAP-SSC-Type
 |  |   +------- Packet Length field set to 31
 |  +------- Identifier field has been incremented to 167
 +------- Code Field set for EAP-Success packet
73 74 6F 70 ; M3="stop"
E69D06BA33DF2799B436D65A348F33840B332810 ; D3=D("stop" | D2 | SK)
//******************* End of the 5th packet *********************
```

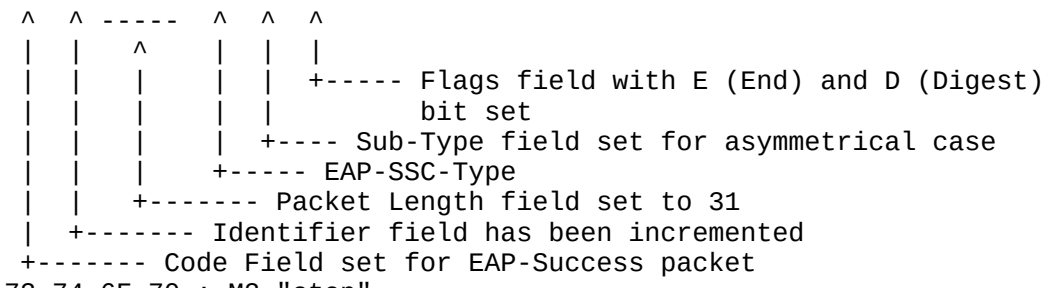
9.2 Traces in an asymmetrical key exchange context

We assume the Authentication Server knows the public key of the Smart Card, and the public key of the Authentication Server is also known by the Smart Card. For these reasons, fields used by certificates C1 and C2 in exchanged packets are empty.

```
//** First Pair-wise-key used by the Authentication Server
//* Value of Modulo1 - Integer 129 bytes
02 81 81
00EE9D84FB3D70CD3CF145BDB8D1D7580BDB917149D44EE09C6E8409853E7D68
5A7C61F840B687EC0F841FEDBCEA6FBBD872783C43CA04AEA56956BD607AAB38
739E629C6FAE2D34B69FFD3D722BE41719CFA5122B50D7821A4FF69DB5E6839D
5938D8D8FD830488342AA5A266A45CD8C1AE32E59B66EE1FFA65DEBD6235824B
21
//* Value of K1public - Integer = 3
02 01 03
//* Value of K1private - Integer 129 bytes
02 81 81
009F13ADFCD3A088D34B83D3D08BE4E55D3D0BA0DBE2DF406849AD5BAE29A8F0
```

```
3C52EBFAD5CF05480A581549289C4A7D3AF6FAD2D7DC031F18F0E47E4051C77A
F6754030B429325864665ECE80839E26AAE039CE642E8253A7E4074BC934D109
8FC5FA3F6D9985251A3123BAB9AEA498F81FE5EE4407195757FED591D09F5D10
CB

//** Second Pair-wise-key used by the Smart Card
//* Value of Modulo2 - Integer 65 bytes
02 41
00B7C2DF803986F6F4DFBA2E104FC5DE0F8DC50ABE713DB9AA2B78387996DCC6
437FFA8B24CD657FAEEE02082EA01553E2DC0A68A5FD5891AAEF78C2489CAB50
C1
//* Value of K2public - Integer = 3
02 01 03
//* Value of K2private - Integer 64 bytes
02 40
7A81EA557BAF4F4DEA7C1EB58A83E95FB3D8B1D44B7E7BC6C7A57AFBB9E8842B
DD5FA9723EC5BF7A9CB387AF255583620B98FE5F0020EE72E24BB429D4BBCACB

//******** Beginning of the 1st packet of the exchange *********
//*********** sent by the Authentication Server (AS) ***********
01 A5 00 2D FF 02 20 ; header
 ^  ^ -----  ^  ^  ^
 |  |   ^    |  |  |
 |  |   |    |  |  +----- Flags field with S (Start) bit set
 |  |   |    |  +---- Sub-Type field set for asymmetrical case
 |  |   |    +----- EAP-SSC-Type
 |  |   +------- Packet Length field set to 45
 |  +------- Identifier field
 +------- Code Field set for EAP-Request packet

02 84 00 00 00 20 ; ASN.1 header of the integer r1

//* Value of r1 on 32 octets (256 bits)
005A9B7B1ABDF0A329B3AB16E5F8933154E33C2C4ADD82F4DD2753257FF62ADC
//******************* End of the 1rst packet ******************

//* Value of r2 on 128 octets (1024 bits)
006696D8F9847CAC6FD072E68E7339B8A96BCD4E7D5E2C2B69CF802F79F584EA
AEB85C19D59986E285CCBF86EE4AEB5B0061909165A0B6E3CDA8AA21704C363B
7475F198E22320CDF3B86F40B46EC879482718C5DF242A72A081E674C763469B
B55E6B5946FF5BF7DB82E22194EC4F4C177C067A980A4B945DED75B0C8B23F19

//* Value of U on 128 bytes (1024 bits) equals to the encryption
//* of r2 with the public key K1public of the AS
7E36D476944C29467915734360D647D6A8923043B727548495A265B7A38CACBE
0CEF55DF16911AA8A63BFB55D5262D14A1D4FC82B0DF011AD61FD243916C4682
A73E647E1269785EECEE414BCFE43660E107D120E30CED09151D884D15B0BA94
17F038955AF4B68621AF0EC3E38DBCCB0827961813B26123FE001DB0E0316211

//* Value of D0 on 20 bytes computed as the digest of the
//* concatenation of fields from Code field to integer value U
//* coded in the packet by the Smart Card
9E7EFE6B9C60428CC61C8798C8F4FE4835BA0861

//* Value of V on 64 bytes (512 bits) equals to the encryption
//* of D0 with the private key K2private of the Smart Card
3A95A34B98F5E009FAE2ECE3F836DFEBB73EEC8B89F733C02F74EBB236AB6151
5D003228F355877C94AFDAAADEC5C47F236F09FE1D8E651FAFE757F064292B73

//******** Beginning of the 2nd packet of the exchange *********
//*************** sent by the Smart Card to the AS *************
02 A5 00 D3 FF 02 00 ; header
 ^  ^ -----  ^  ^  ^
 |  |   ^    |  |  |
```

```
 |  |   |    |  |  +----- Flags field
 |  |   |    |  +---- Sub-Type field set for asymmetrical case
 |  |   |    +----- EAP-SSC-Type
 |  |   +------- Packet Length field set to 211
 |  +------- Identifier field
 +------- Code Field set for EAP-Response packet

02 84 00 00 00 80 ; ASN.1 header of the integer U value
                  ; coded on the 128 bytes below
7E36D476944C29467915734360D647D6A8923043B727548495A265B7A38CACBE
0CEF55DF16911AA8A63BFB55D5262D14A1D4FC82B0DF011AD61FD243916C4682
A73E647E1269785EECEE414BCFE43660E107D120E30CED09151D884D15B0BA94
17F038955AF4B68621AF0EC3E38DBCCB0827961813B26123FE001DB0E0316211

02 84 00 00 00 40; ASN.1 header of the integer V value
                 ; coded on the 64 bytes below
3A95A34B98F5E009FAE2ECE3F836DFEBB73EEC8B89F733C02F74EBB236AB6151
5D003228F355877C94AFDAAADEC5C47F236F09FE1D8E651FAFE757F064292B73
//******************** End of the 2nd packet ********************
//* value of the Session's Key SK
3B4C5E8CD72D723A6CC971612DFFED0EB1E8B514

//* value of D1=D("hello" | SK)
772EC3BD82C07C9A8F06FE006ED779EA7AAB8B77

//******** Beginning of the 3rd packet of the exchange *********
//************** sent by the AS to the Smart Card **************
01 A6 00 20 FF 02 08
 ^  ^ -----  ^  ^  ^
 |  |   ^    |  |  |
 |  |   |    |  |  +----- Flags field with D (Digest) bit set
 |  |   |    |  +---- Sub-Type field set for asymmetrical case
 |  |   |    +----- EAP-SSC-Type
 |  |   +------- Packet Length field set to 32
 |  +------- Identifier field has been incremented to 166
 +------- Code Field set for EAP-Request packet
68 65 6C 6C 6F ; M1="hello"
772EC3BD82C07C9A8F06FE006ED779EA7AAB8B77 ; D1=D("hello" | SK)
//******************** End of the 3rd packet ********************

//* value of D2=D("world" | D1 | SK)
CB2A67FAEB44BBC841E99ECAD6C8B25B2FCB3122

//******** Beginning of the 4th packet of the exchange *********
//************** sent by the Smart Card to the AS **************
02 A6 00 20 FF 02 08
 ^  ^ -----  ^  ^  ^
 |  |   ^    |  |  |
 |  |   |    |  |  +----- Flags field with D (Digest) bit set
 |  |   |    |  +---- Sub-Type field set for asymmetrical case
 |  |   |    +----- EAP-SSC-Type
 |  |   +------- Packet Length field set to 32
 |  +------- Identifier field is the same as for request packet
 +------- Code Field set for EAP-Response packet
77 6F 72 6C 64 ; M2="world"
CB2A67FAEB44BBC841E99ECAD6C8B25B2FCB3122 ; D2=D("world" | D1 | SK)
//******************** End of the 4th packet ********************

//* value of D3=D("stop" | D2 | SK)
D5ACB12A9F74B2E09B5CB1788D1EBE8AE6E8027C

//******** Beginning of the 5th packet of the exchange *********
//************** sent by the AS (EAP-Success/End) **************
03 A7 00 1F FF 02 18
```




```
73 74 6F 70 ; M3="stop"
D5ACB12A9F74B2E09B5CB1788D1EBE8AE6E8027C ; D3=D("stop" | D2 | SK)
//******************* End of the 4th packet ********************
```