Le protocole EAP-SSC

Mesmin T. Dandjinou* - Pascal Urien**

* Ecole supérieure d'informatique
Université polytechnique de Bobo-Dioulasso
01 BP 1091 Bobo-Dioulasso
BURKINA FASO
Mesmin.Dandjinou@voila.fr

** Ecole nationale supérieure des télécommunications de Paris
46, rue Barrault
75013 Paris
FRANCE
Pascal.Urien@enst.fr

RÉSUMÉ. Avec l'usage de plus en plus répandu de terminaux intelligents mobiles et des réseaux sans fil peu coûteux, il est de nos jours aisé de s'introduire dans un réseau local. Aussi la sécurité est un aspect important à maîtriser lors de la configuration et de la maintenance de ces réseaux. Or les progrès technologiques de la carte à puce rendent actuellement envisageable l'utilisation de leurs capacités de calcul et de stockage dans le processus d'authentification. Dans cet article, nous proposons un protocole léger, robuste et sécurisé qui pourrait être d'une part employé aussi bien dans un contexte de cryptographie symétrique qu'asymétrique, et d'autre part implémenté comme une méthode EAP sur une carte à puce, de manière à mettre en place un canal sécurisé d'échanges pouvant par la suite servir à la transmission de messages de management comme par exemple ceux requis pour gérer un environnement LDAP installé sur les cartes à puce.

ABSTRACT. The emergence of mobile intelligent terminals and cheaper wireless networks makes easier nowadays to enter in a network. For this reason, security is an important factor to consider during the configuration and the maintenance of networks. With the evolution of the smart cards technologies it becomes more and more feasible to use their capabilities of storage and computation for authentication process. In this paper we propose a light weighted but robust secure protocol usable in both symmetrical and asymmetrical encryption contexts that can be implemented like an EAP method in smart cards to set up a secured channel that can be used after to convey management frames for by example an embedded LDAP repository on the smartcards.

MOTS-CLÉS : Réseaux, protocole EAP, cartes à puce, cryptographie.

KEYWORDS: Networks, EAP protocol, smartcards, cryptography.

1. Introduction

Dans un réseau sans fil, la sécurité ne peut pas être assurée comme dans un réseau filaire entre quatre murs, à cause des liens radio employés. C'est pour y remédier qu'ont été faites les spécifications IEEE 802.11 introduisant le protocole WEP (Wired Equivalent Privacy). Malheureusement beaucoup de failles ont depuis lors été recensées dans la manière de réaliser l'authentification, la signature des paquets et la confidentialité au niveau du protocole WEP [3] [8]. Dans l'optique d'une meilleure résolution de ces problèmes, le standard IEEE 802.1X basé sur une authentification de port pour les utilisateurs et les périphériques (dans les réseaux filaires et sans fil), a proposé une architecture plus robuste et un processus d'authentification offrant au-dessus de la couche protocolaire EAP (Extensible Authentication Protocol) [4] une large gamme de méthodes d'authentification non négociées à l'avance, en attendant l'arrivée de nouvelles spécifications formulées par le groupe de travail IEEE.11i.

Presque en même temps, les technologies des cartes à puce ont fait de grands bonds en avant, leur conférant aujourd'hui des caractéristiques qui les font figurer parmi les meilleurs périphériques résistant à la casse et utilisés pour sécuriser l'authentification des utilisateurs, en association avec le protocole EAP [11].

S'il existe des méthodes d'authentification robustes, leurs exigences en termes de puissance de calcul et de capacité de stockage constituent des obstacles pour leur implémentation sur des cartes à puce. C'est en cela que le protocole EAP-SSC (EAP Secured Smartcard Channel), plus léger, pourrait représenter une solution intéressante.

2. Contexte d'utilisation du protocole EAP-SSC

Le lieu d'utilisation du protocole EAP-SSC correspond à celui décrit par le standard IEEE 802.1X [9] pour le contrôle d'accès aux réseaux filaires et sans fil. C'est là qu'est défini le standard EAP améliorant la sécurité des technologies 802.11 grâce au cadre de réalisation des processus d'authentification mutuelle propres aux applications. Trois acteurs majeurs se reconnaissent dans cet environnement : le serveur d'authentification (traduction de « *authentication server* »), le client à authentifier (traduction du terme « *supplicant* ») et l'authentifieur (traduction du terme « *authenticator* »).

Le serveur d'authentification permet d'authentifier et d'autoriser la connexion d'un utilisateur donné à un port du réseau local. En fait, c'est au niveau de ce serveur central que sont conservées les informations concernant les utilisateurs reconnus au niveau du réseau local et les privilèges (profils) accordés à chacun d'eux.

Le client à authentifier correspond à tout équipement employé par un utilisateur pour accéder à un réseau local, aussi bien dans un environnement filaire que sans fil.

Enfin, l'authentifieur qui est le point d'entrée ou d'accès au réseau local devant authentifier et vérifier les droits reconnus à l'utilisateur demandant à se connecter ; c'est l'intermédiaire obligatoire entre le client à authentifier et le serveur d'authentification.

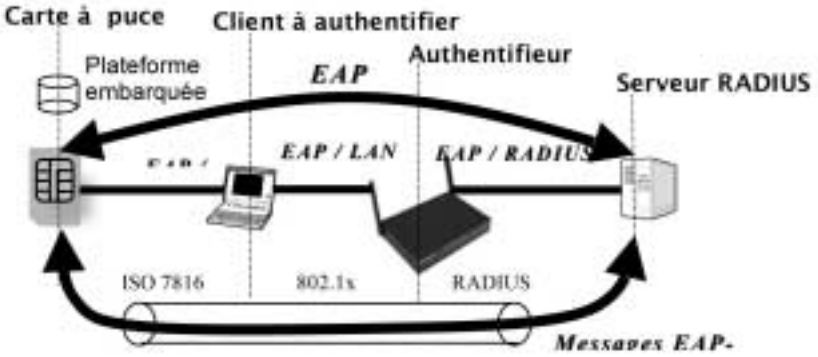

Figure 1. *Contexte d'emploi du protocole EAP-SSC*

Le protocole EAP-SSC est conçu pour réaliser un canal sécurisé d'échanges entre une carte à puce et un serveur d'authentification. Il s'emploie comme l'une des méthodes d'authentification supportées par EAP.

En effet, même s'il est possible de stocker en partie un client à authentifier sur une carte à puce, ce n'est encore pas le cas pour le gros et complexe protocole qu'est EAP-TLS [1] [5]. C'est cette situation qui est à l'origine du protocole EAP-SSC comparable à un EAP-TLS allégé. Dans un contexte d'accès sans fil à un réseau local dans lequel un canal physique sécurisé existe entre l'authentifieur et le serveur d'authentification (comme RADIUS), EAP-SSC permet d'établir un canal sûr entre un client hébergé partiellement sur une carte à puce [12] [13], et le serveur d'authentification (fig. 1).

3. Principales caractéristiques du protocole EAP-SSC

Le protocole EAP-SSC fondé sur le protocole EAP est simple, robuste et utilisable dans un contexte de cryptographie aussi bien symétrique qu'asymétrique où une authentification mutuelle est réalisée avant tout échange entre la carte à puce et le serveur d'authentification sur la base d'une clé de session partagée mais non distribuée servant par la suite à dériver d'autres clés et paramètres de chiffrement non négociés. Il garantit l'intégrité des messages, leur confidentialité, la non répudiation et la protection contre les attaques par hôte interposé ou par rejeu.

3.1. Format du paquet EAP-SSC

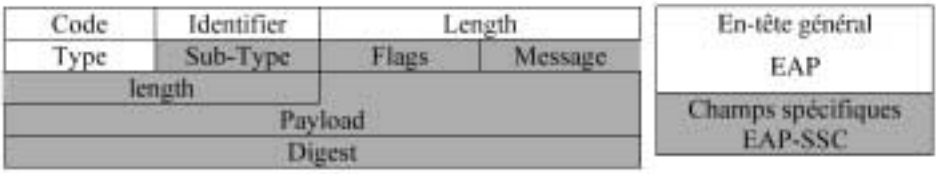

Figure 2. *Format du paquet EAP-SSC*

A la figure 2 est illustré un paquet EAP-SSC encapsulé dans un autre paquet EAP classique. Les champs *Code*, *Identifier*, *Length* et *Type* sont décrits selon le RFC 2284 [4]. L'emploi de EAP-SSC nécessite l'attribution par l'IANA d'une nouvelle valeur pour le champ *Type* qui permettra de distinguer ce nouveau protocole de ceux existant.

Le champ *Sub-Type* long d'un byte distingue plusieurs familles de messages ; actuellement *Sub-Type = 1* correspond au choix du contexte de chiffrement symétrique, et *Sub-Type = 2* au modèle cryptographique asymétrique. Le champ *Flags* d'un octet complète le champ *Sub-Type* ; ainsi pour les valeurs 1 ou 2 du *Sub-Type*, les bits du *Flags* ont la signification donnée à la figure 3, 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 dans l'EAP-TLS. Le bit E (*End*) à 1 dans un paquet émis par le serveur d'authentification marque la différence avec les autres messages. Le bit D (*Digest*) 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.

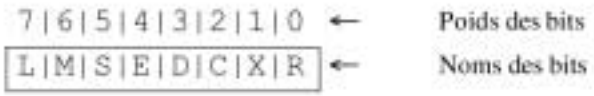

Figure 3. *Format du champ Flags pour Sub-Type=1 ou 2*

Message Length est un champ de trois octets figurant dans le message EAP-SSC lorsque le bit L est à 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 paquet EAP-SSC quand le bit D du champ *Flags* vaut 1 ; il a 160 bits et correspond au condensé calculé, par défaut selon l'algorithme *Secure Hash Algorithm* (*SHA-1*) [6], sur une entrée résultant de la concaténation d'une liste précise de champs du message.

3.2. Authentification mutuelle au sein du protocole EAP-SSC

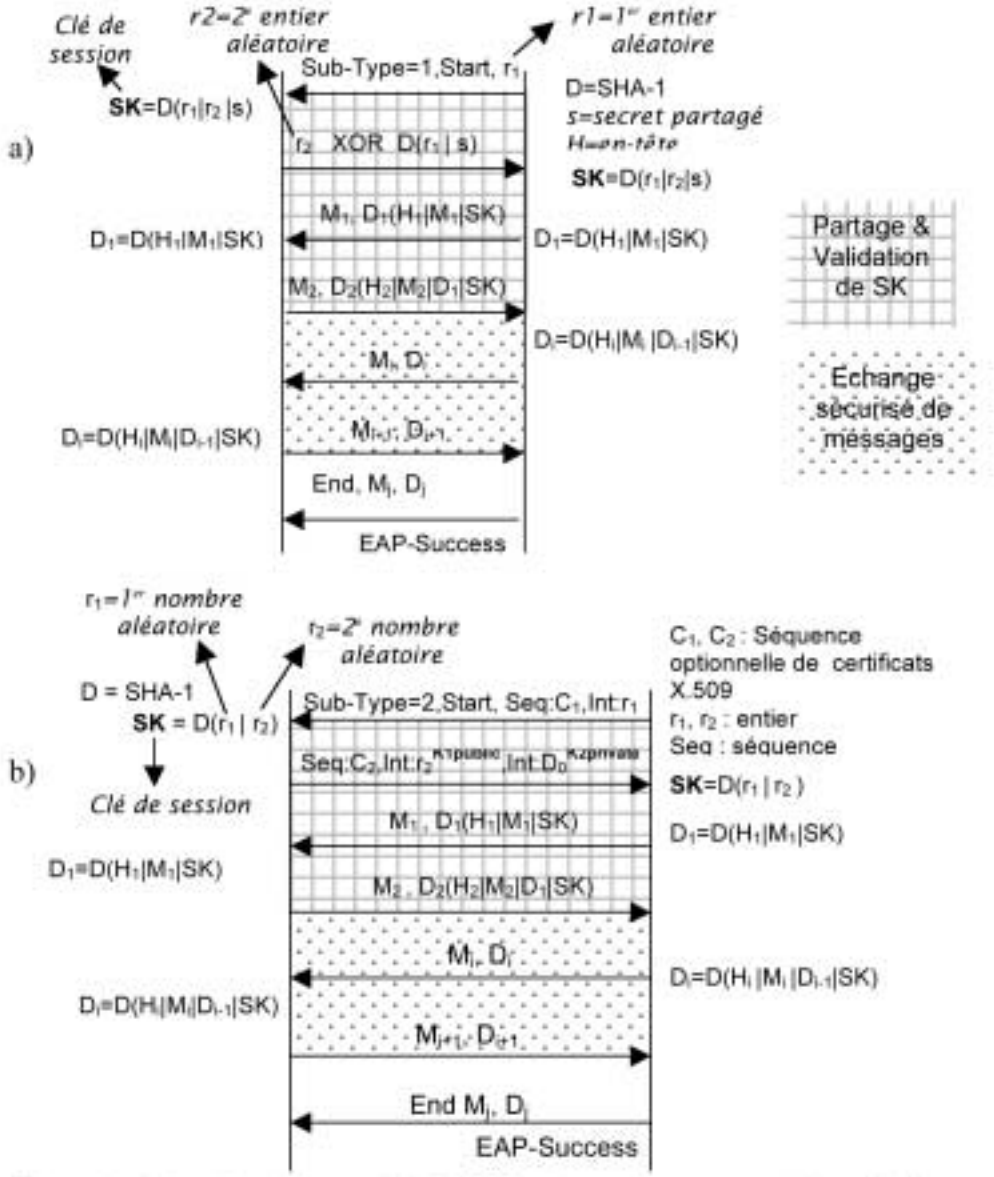

Figure 4. *Authentification mutuelle EAP-SSC : a) contexte cryptographique à clé symétrique b) contexte cryptographique asymétrique*

Une authentification mutuelle entre la carte à puce et le serveur d'authentification a lieu dès que le serveur d'authentification reçoit de l'authentifieur le paquet EAP-Response/Identity dans lequel est supposé 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**, suivie ensuite de celle d'**échange sécurisé de messages**. Une illustration de ces phases dans les deux contextes cryptographiques est fournie à travers les figures 4 a) et b).

3.2.1. Phase de partage de clé de session EAP-SSC dans un contexte cryptographique à clé symétrique

Quand le serveur d'authentification reçoit de l'authentifieur un paquet EAP-Response/Identity valide, il génère un nombre entier aléatoire positif r_1 de 160 bits de longueur (le bit de plus fort poids de r_1 étant toujours à 0). Ce nombre r_1 est empaqueté en clair, les octets de poids faible d'abord, dans un bloc EAPOR-Request/EAP-SSC/Start envoyé ensuite au client à authentifier.

L'authentifieur 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 client à authentifier, enverra en résultat à ce dernier un paquet EAPOL-Request/EAP-SSC/Start. A la réception de ce paquet, le client à authentifier récupère l'entier positif aléatoire r_1 envoyé par le serveur d'authentification.

Tout comme le serveur d'authentification, le client à authentifier lors de la préparation de sa réponse, va générer aléatoirement un nombre entier positif r_2 de 160 bits de longueur (avec ici aussi le bit de plus fort poids de r_2 à 0) qu'il emploiera ensuite dans le calcul de la valeur $Z=(r_2 \text{ XOR } SHA\text{-}1(r_1 \mid s))$, où s correspond au secret partagé par le serveur d'authentification et le client à authentifier, valeur qui dans le cas du client à authentifier est supposée conservée au niveau de la carte à puce. La valeur Z qui correspond en fait au résultat du chiffrement du nombre aléatoire r_2 que vient de générer le client à authentifier, est à son tour empaquetée, les octets de poids faible d'abord, dans un paquet EAPOL-Response/EAP-SSC transmis au serveur d'authentification via l'authentifieur. A partir de cet instant, le client à authentifier est en mesure de calculer la clé de session $SK=SHA\text{-}1(r_1 \mid r_2 \mid s)$ du fait de la possession du triplet (r_1, r_2, 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 r_1 et s, le serveur d'authentification est en mesure de calculer le condensé $SHA\text{-}1(r_1 \mid s)$, ce qui lui ouvre la voie au décryptage de la valeur de r_2 cachée dans Z. Dès lors, le serveur d'authentification à son tour dispose du triplet (r_1, r_2, s) qui lui permet de calculer à son niveau la clé de session SK, tout comme le client à authentifier : 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 présenterons un peu plus loin.

3.2.2. Phase de partage de clé de session EAP-SSC dans un contexte cryptographique à clé asymétrique

Dans le contexte cryptographique à clé publique, on fait plutôt usage de certificats qui correspondent à des documents électroniques renfermant des clés publiques et d'autres données complémentaires permettant la réalisation des opérations de chiffrement et de déchiffrement. Quand le serveur d'authentification dispose du certificat du client à authentifier et que le client à authentifier dispose aussi du certificat du serveur d'authentification, l'échange des certificats peut être court-circuitée ; et dans ce cas les variables C1 et C2 désignant les certificats échangés sont à vide.

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 r_1 dont il est seul à déterminer la longueur variable dans le temps (l'octet de plus fort poids étant toujours à 0). r_1 et la séquence optionnelle de certificats nommée C1 appartenant au serveur d'authentification sont représentés au format BER ASN.1 [10] et empaquetés en clair dans un paquet EAPOR-Request/EAP-SSC/Start expédié au client à authentifier via l'authentifieur. Dès que ce paquet parvient au client à authentifier, ce dernier en extrait la valeur aléatoire positive r_1 ainsi que la chaîne optionnelle de certificats C1, ce qui lui donne de disposer d'une part de la clé publique désignée par K1public et d'autre part de la base du modulo appelée Modulo1 de l'expéditeur du paquet reçu.

Le client à authentifier, en réponse au message reçu, génère à son tour un nombre aléatoire positif r_2 de longueur quelconque (mais avec toujours l'octet de plus fort poids à 0) dont il se sert pour calculer deux valeurs U et V telles que $U=((r_2$ ** K1public) MOD Modulo1) et $V=((D_0$ ** K2private) MOD Modulo2), avec D_0=SHA-1(Code | Identifier | Length | Type | Sub-Type | Flags | C2 | U). En fait D_0 correspond au condensé calculé sur la concaténation de l'ensemble des champs précédant le champ *Digest* dans le paquet que le client à authentifier se prépare à envoyer en réponse au serveur d'authentification. La valeur de U correspond au chiffrement du nombre entier aléatoire r_2 avec la clé publique du serveur d'authentification, tandis que celle de V correspond au cryptage du condensé D_0 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 emballés dans un paquet EAPOL-Response/EAP-SSC acheminé au serveur d'authentification via l'authentifieur. Dès lors, le client à authentifier est en mesure de calculer la clé de session $SK=SHA\text{-}1(r_1 | r_2)$.

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 D_0 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 r_2 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 (r_1, r_2), le serveur d'authentification est lui aussi en mesure de calculer la clé de session SK=SHA-1($r_1 | r_2$), mettant ainsi un terme à cette phase.

3.2.3. 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 client à authentifier, elle a comme 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 client à authentifier et le serveur d'authentification. A l'initiative du serveur d'authentification, un paquet EAPOR-Request/EAP-SSC est envoyé via l'authentifieur au client à authentifier. La particularité de ce paquet est de renfermer un message désigné par M_1 pouvant même être vide mais systématiquement complété par un condensé de message D_1=SHA-1(Header(M_1) | M_1 | SK), avec Header(M_1) correspondant à l'en-tête d'un paquet donné contenant le message M_1, et constitué par la suite de champs précédant le champ *Payload* (voir figure 2) dans le paquet. Dans le cas présent où le paquet achemine le message M_1, l'en-tête Header(M_1) correspondra à la concaténation des champs *Code*, *Identifier*, *Length*, *Type*, *Sub-Type* et *Flags*.

A la réception de ce paquet qui a transité par l'authentifieur, le client à authentifier en extrait le condensé du message D_1 provenant du serveur d'authentification. Il procède ensuite à la comparaison du condensé D_1 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 D_2 selon la formule D_2=SHA-1(Header(M_2) | M_2 | D_1 | SK), où M_2 correspond au message (pouvant être de longueur nulle) renvoyé en réponse à M_1. M_2 et D_2 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 D_2 (reçu). Disposant de D_1 et de SK, le serveur d'authentification calcule à son tour D_2 (local) et le compare au condensé D_2 (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.

3.3. 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 client à authentifier est joint un condensé de message D_i calculé à partir de la clé de session SK, du message M_i à expédier et du condensé D_{i-1} du message reçu à la précédente étape : D_1=SHA-1(Header(M_1)|M_1|SK) et D_i=SHA-1(Header(M_i) | M_i | D_{i-1} | SK) pour i > 1, avec SK calculé à partir soit du triplet (r_1, r_2, s) dans un contexte cryptographique à clé secrète, soit du couple (r_1, r_2) quand on se retrouve dans l'environnement cryptographique asymétrique.

Au moins 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 la réponse qui provient du client à authentifier et qui peut 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 client à authentifier. 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.

3.4. Chiffrement réalisé 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 le cryptage du message est indiquée par l'utilisation du bit C du champ *Flags*. Par défaut l'algorithme de cryptage employé est le Triple Data Encryption Algorithm (3DES) correspondant au standard décrit dans ANSI X9.52 [7] fonctionnant selon le schéma EDE (Encode-Decode-Encode) en mode CBC (Cipher Block Chaining) avec l'option 2 de clés (K_1, K_2, K_3) où K_1 et K_2 sont des clés indépendantes tandis que K_3=K_1.

4. Conclusion et travaux futurs

Des simulations ont été faites et montrent que les calculs envisagés sont bien réalisables sur une carte à puce. C'est une première étape dans la conception d'un protocole usant d'algorithmes cryptographiques tels que SHA-1 et 3DES pour fournir un autre moyen pour réaliser un canal de communication sécurisé entre une carte à puce et un serveur d'authentification. Puis suivra une implémentation expérimentale. Comme tous les aspects du sans fil n'ont pas été pris en compte, les travaux futurs doivent améliorer les mécanismes mis en œuvre par le protocole et prendre en compte des algorithmes et mécanismes plus efficaces comme AES, SHA-256, la dérivation de clés, la probable

évolution de EAP. L'utilisation première prévue du canal sécurisé via le protocole EAP-SSC est l'acheminement des trames de gestion d'une plate-forme LDAP embarquée sur une carte à puce ; ce sera l'objet de nos travaux futurs.

6. Bibliographie

[1] ABOBA B., SIMON D., *PPP EAP TLS Authentication Protocol*, RFC 2716, October 1999, http://www.ietf.org/rfc/08.txt

[2] ANDERSSON H., JOSEFSSON S., ZORN G., SIMON D., PALEKAR A., *Protected EAP Protocol (PEAP)*, draft-josefsson-pppext-eap-tls-eap-05.txt, work-in-progress, September 2002. (INFORMATIVE)

[3] ARBAUGH W. A., SHANKAR N., WAN J., *Your 802.11 Wireless Network has No Clothes*, Department of Computer Science, University of Maryland, March 2001.

[4] BLUNK L., VOLLBRECHT J., *PPP Extensible Authentication Protocol (EAP)*, RFC 2284, March 1998.

[5] DIERKS T., ALLEN C., *The TLS Protocol Version 1.0*, RFC 2246, January 1999, http://www.ietf.org/rfc/rfc2246.txt

[6] FIPS PUB 180-1, *Secure Hash Standard*, April 1995, http://www.itl.nist.gov/fipspub/fip180-1.htm.

[7] FIPS PUB 46-3, *Data Encryption Standard (DES)*, October 1999, http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf.

[8] FLUHRER S., MANTIN I., SHAMIR A., *Weakness in the key scheduling algorithm of RC4*, 8th Annual Workshop on Selected Areas in Cryptography, August 2001.

[9] IEEE STD 802.1X-2001, *IEEE standard for local and metropolitan area networks - Port-based network access control*, LAN/MAN Standards Committee of the IEEE Computer Society, USA, 2001.

[10] ITU-T Rec. X.690 (2002) | ISO/IEC 8825-1:2002, *Information technology – ASN.1 encoding rules - Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*, 2002.

[11] URIEN P., FARRUGIA A. J., GROOT M., PUJOLLE G., ABELLAN J., *EAP-Support in smartcard*, draft-urien-03-02.txt, work-in-progress, January 2004.

[12] URIEN P., LOUTREL M., LU K., *Introducing Smart Cards for Wireless LAN Security*, 10th International Conference on Telecommunication Systems, Monterey, California, October 3-6 2002.

[13] URIEN P., LOUTREL M., *The EAP Smart Card, a tamper resistant device dedicated to 802.11 wireless networks*, ASWN 2003, July 2003.