EAP-TLS sur carte à puce, une PKI de poche, applications et perspectives

Pascal Urien, ENST Paris

Introduction

Le début du 21ième siècle est marqué par l'émergence des réseaux IP sans fil, qui tisseront probablement les toiles de communication de l'informatique enfouie. Le formidable succès du réseau GSM a constitué une étape décisive quant au développement de services liés à la mobilité. En particulier l'introduction des cartes SIM est un point de rupture avec les mécanismes classiques de sécurité, basés sur des mots de passe, déployés dans l'Internet de première génération.

La norme Wi-Fi (IEEE 802.11) a été finalisée en 99; malgré l'extraordinaire potentiel économique de cette technologie son déploiement a été freiné voire interdit en raison de problèmes de sécurité. C'est un évènement remarquable dans l'histoire de la télématique, pour la première fois la vague IP a été endiguée par un manque de sûreté, alors que le succès même de l'Internet provenait pour partie d'une quasi absence de sécurité, tant au niveau structurel que protocolaire. A titre d'illustration nous remarquerons que bon nombre de standards IP (RFCs) s'appuient encore sur la fonction MD5, pour laquelle de nouvelles collisions furent démontrées lors de la conférence CRYPTO 2 004.

Après cinq années d'efforts, les comités IEEE et IETF ont bâti une architecture sécurisée reposant sur un socle de normes telles que IEEE 802.1x (2001) et IEEE 802.11i (2004). Le protocole EAP (*Extensible Authentication Protocol, [3]*) joue un rôle privilégié dans ce modèle, à l'aide de ses cinq octets d'en tête, il définit un cadre générique et flexible pour le transport de protocoles d'authentification, répondant à de multiples paradigmes. De nombreux standards, en quête de mécanismes d'authentification supportent EAP, par exemple l'IEEE 802.16e (WiMax) ou IKEv2, la nouvelle version du protocole d'échange de clé d'Internet, utilisé pour l'ouverture de tunnels IPSEC.

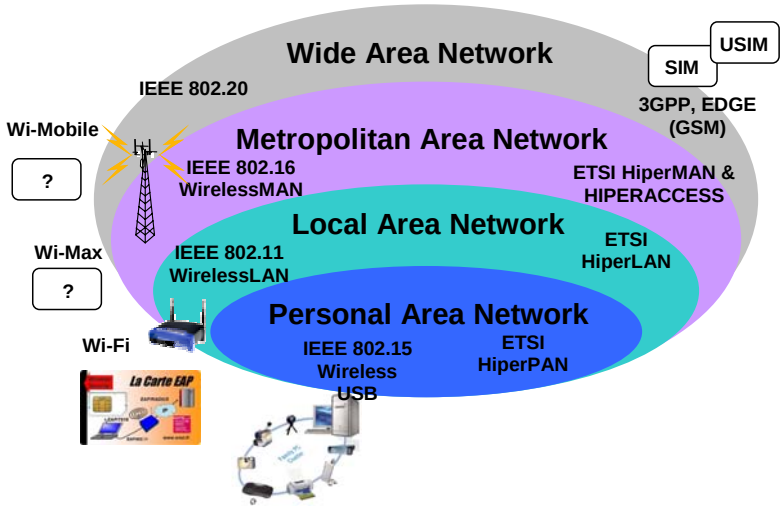

Figure 1 : IP sans fil et cartes à puce

Taxonomie des mécanismes d'authentification.

Nous classerons les méthodes d'authentification en trois catégories, symétriques (secrets partagés) asymétriques (basés sur RSA en règle générale) et tunnels.

Mécanismes symétriques

Nous divisons les méthodes symétriques en trois classes

- Les mots de passe. Un mot de passe est une suite de caractères qui peut être facilement mémorisé par un être humain. Il ne s'agit pas d'un nombre aléatoire mais au contraire d'une concaténation de mots, de chiffres et de signes. De tels secrets peuvent être devinés à l'aide d'une attaque par

dictionnaire, c'est à dire un programme utilisant une base de donnée pour la génération de ces valeurs. Ainsi les méthodes EAP-MSCHAPv2 ou LEAP reposent sur la *clé NT*, c'est à dire l'empreinte MD4 (soit 16 octets) d'un mot de passe.

- Les secrets partagés (PSK, Pre-Shared-Key). Il s'agit en fait d'un nombre aléatoire, dont la taille est l'ordre de 128 à 160 bits. Un cerveau humain éprouve beaucoup de difficultés à mémoriser ces informations. Le stockage sécurisé du secret est réalisé par exemple par le système d'exploitation d'un ordinateur personnel ou par une carte à puce.

- Le mode *provisioning*. Dans les réseaux GSM ou UMTS une base de donnée centralisée (*Host Location Register*) gère les comptes utilisateurs, en particulier leur PSK. Afin d'éviter des interrogations fréquentes les méthodes d'authentification (EAP-SIM, EAP-AKA…) sont conçues de telle manière que le site central puisse produire des vecteurs d'authentification, re-utilisables par des agents de confiance (tels que les *Visitor Location Register* par exemple).

Mécanismes Asymétriques

Ces procédures sont basées sur des algorithmes tels que RSA ou Diffie-Hellman. Le protocole SSL/TLS [1] est généralement utilisé pour le transport de ces échanges. La RFC 2716 [2] (EAP-TLS) décrit l'encapsulation transparente de TLS dans des messages EAP.

Les tunnels

Ainsi que nous l'avons souligné précédemment les méthodes d'authentification basées sur des mots de passe, sont sujettes à des attaques par dictionnaire. Les protocoles MSCHAP ou LEAP, assurent une protection jugée raisonnable dans des environnements sures (par exemple des intranets), mais ne sont plus adaptés lors d'une mise en œuvre dans un milieu hostile, tel que IP sans fil, abritant potentiellement de nombreuses oreilles électroniques indiscrètes.

Les tunnels, s'appuyant fréquemment sur la technologie SSL/TLS, protégent un scénario d'authentification grâce au chiffrement des données échangées. Il existe aujourd'hui de nombreuses propositions (PEAP défini par *Microsoft*, FAST-EAP spécifié par *Cisco*, TTLS proposé par *Funk Software*) devenues nécessaires en raison des nombreux logiciels disponibles sur le WEB qui cassent les protocoles à base de mot de passe.

La carte à puce EAP

Le concept de carte à puce EAP est né à la fin de l'année 2002, c'est à dire pratiquement avec la standardisation de l'architecture IEEE 802.1x. L'idée de base est de traiter les messages EAP dans la puce sécurisée, chaque requête est analysée par la carte, qui délivre la réponse induite. L'interface fonctionnelle (c'est à dire le jeu de commande ISO 7816), est décrite par un *Draft Internet* [4].

La carte EAP a obtenu le *Sésame de la Meilleure Innovation* lors du salon cartes 2003 à Paris (la plus grande manifestation internationale de ce secteur d'activités) et un *Innovation Breakthrough Award* lors du salon CardTech/SecureTech de Washington en 2004.

De manière pratique, c'est une application écrite en Java (un *applet javacard*), chargée et exécutée par tout système d'exploitation (carte) compatible avec la norme javacard (JC2.1 ou JC2.2). L'usage de machine virtuelle java introduit la notion de portabilité («*write once, run everywhere*») et de flexibilité, puisque l'application est disponible pour les multiples plateformes disponibles sur le marché.

EAP-TLS sur carte à puce

Par opposition aux systèmes centralisés, TLS introduit la notion de délégation, ce qui facilite la résolution du délicat problème du *roaming*. Le client du réseau et un serveur d'authentification partagent une même autorité de certification (CA) , qui crée une classe de confiance basée sur une relation R (R signifiant= «*fait confiance à*»).

(Client R CA) ET(Serveur R CA) => (Client R Serveur)

Il est également possible de restreindre TLS à un mode symétrique, qualifié de reprise de session (ou *session resume*), utilisant un identifiant de session (*Session-ID*, analogue à un login) et un secret maître (*MasterSecret*) similaire à une clé PSK.

Examinons brièvement l'architecture nécessaire à un système Microsoft pour déployer le protocole EAP-TLS. Une entité nommée RAS (*Remote Access Service*) gère les connexions de l'hôte aux fournisseurs de service IP. Les messages EAP-TLS sont traités par une bibliothèque dynamique spécifique (RASTLS.DLL). Cette dernière consomme les services CAPI (Crypto API) pour l'invocation des fonctions cryptographiques (empreinte, chiffrement, …). Un composant du système, le magasin de certificats (*Certificats Store*) stocke les différents certificats X509 de l'utilisateur ou des

autorités de certification. Lorsque un certificat utilise une clé privée, un attribut du magasin pointe sur un CSP (*Crypto Service Provider*), une bibliothèque identifiée par son nom. Ce fournisseur de services cryptographiques utilise une clé privée embarquée dans une carte à puce, ou protégée par le DAPI (*Data Protection API*) un composant logiciel qui chiffre les données critiques à l'aide de clés déduites d'une *clé NT*, c'est à dire du mot de passe du compte utilisateur.

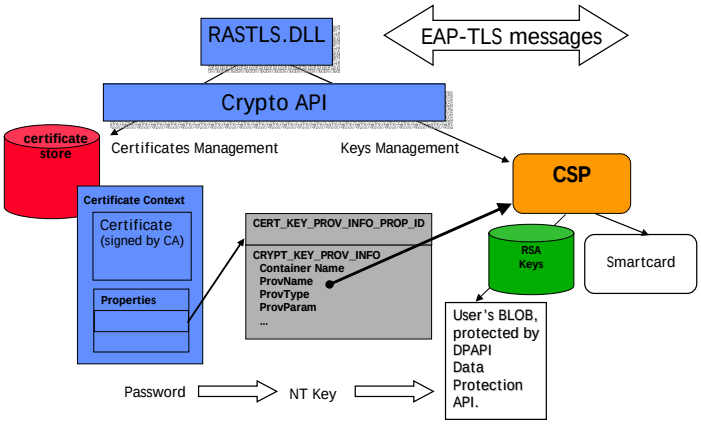

Figure 2 : Architecture EAP-TLS sous Windows.

La gestion du protocole est donc répartie de manière complexe dans l'ordinateur personnel, les possibilités d'attaques sont multiples et finalement la sécurité repose sur un mot de passe. De surcroît le système est configuré avec plusieurs certificats, ce qui limite considérablement l'indépendance de l'utilisateur vis à vis du terminal employé.

Le carte EAP-TLS [5] embarque tous les composants nécessaires au protocole, et comporte en particulier

- Un moteur EAP, qui assure le routage des messages EAP vers les différentes méthodes d'authentification supportées par la carte.

- Une méthode EAP-TLS, qui gère les mécanismes de fragmentation re-assemblage requis par le transport de TLS sur EAP

- Une couche TLS. Le protocole *Handshake* prend en charge les mécanismes d'authentification, le *Record Layer* réalise le chiffrement et l'intégrité des données sécurisées par le tunnel TLS.

- Un magasin de certificats.

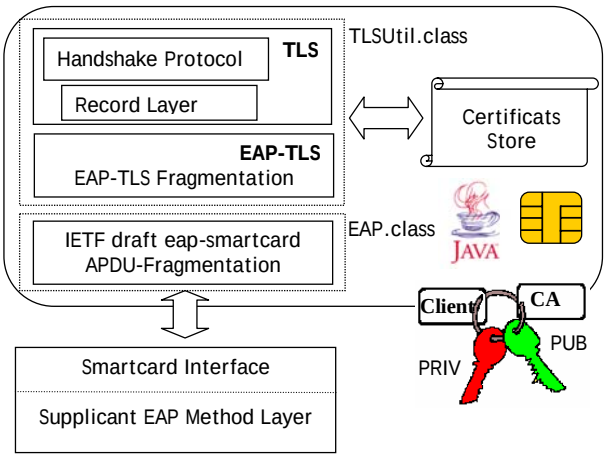

Figure 3 : La carte EAP-TLS

Le terminal réalise un relais passif des messages EAP-TLS, et ne contient aucune information relative à l'utilisateur.

L'applet EAP-TLS.

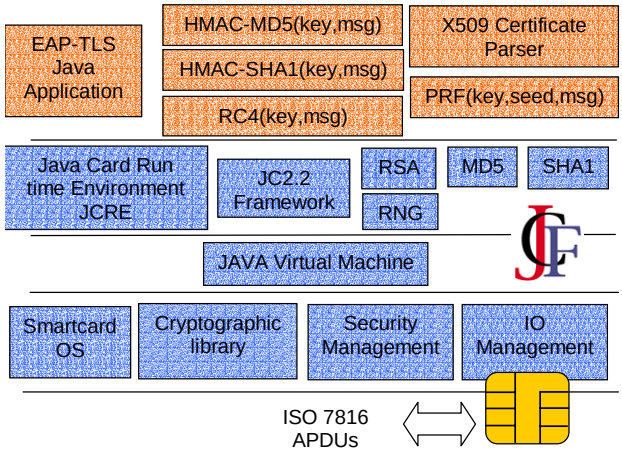

Figure 4 : Architecture java de la carte EAP-TLS

Peu de publications ont étudié de manière approfondie les performances requises par le protocole SSL/TLS. Pour notre part nous avons constaté que côté client, il était possible d'obtenir une implémentation fonctionnelle et

inter opérable, avec un coût d'environ 20 Ko de *code byte* java. Le langage javacard 2.2 supporte un paquetage cryptographique relativement complet, incluant des méthodes pour le calcul des fonctions RSA, MD5 et SHA1. Cependant des parties additives ont été développées en java, telles que les procédures HMAC-MD5, HMAC-SHA1, RC4, fonction PRF(*Pseudo Random Function*) importée de la RFC TLS, et les analyses de certificats X509.

Rappel des échanges TLS.

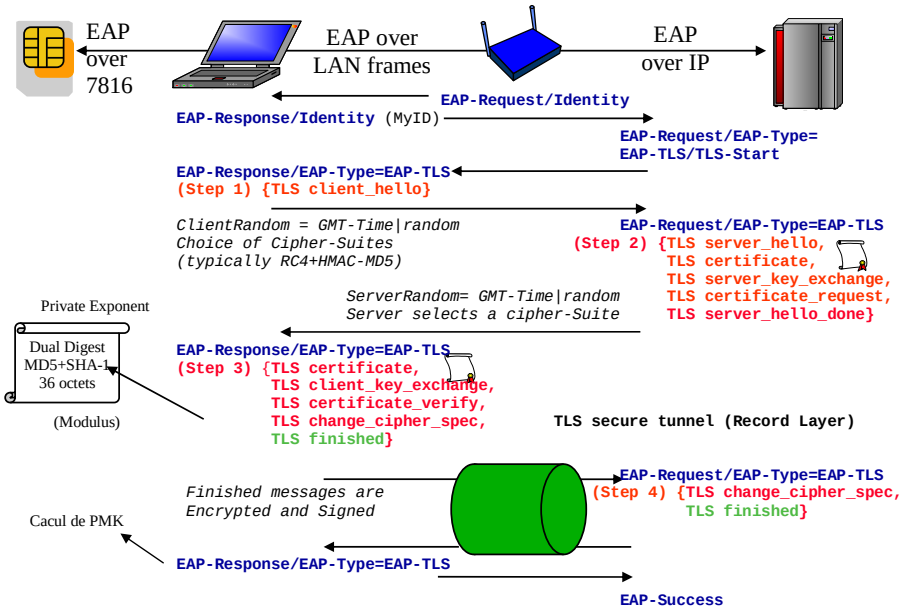

Figure 5: Dialogue TLS

Une authentification EAP-TLS se déroule en 4 étapes

- Etape 1. Le client reçoit un ordre *EAP-TLS/Start* et délivre en retour le message TLS *Client-Hello*.

- Etape 2. Le serveur répond par le message *Server-Hello*, qui inclut ses certificats , ainsi qu'une demande certificat côté client.

- Etape 3. Le client analyse les informations fournies par le serveur, il vérifie sa chaîne de certificats, détermine un secret maître (*Master Secret*) signe avec sa clé RSA privée les messages échangés, et réalise une suite de calculs cryptographiques permettant d'établir les clés utilisées par le tunnel TLS

(*Record Layer*). Il transmet au serveur son certificat et le secret maître chiffrée par la clé RSA publique du serveur.

- Etape 4. Le serveur prouve sa connaissance du secret maître. Lors de la réception de cet ultime message, le client calcule la clé de session PMK (*Pairwise Master Key*) qui sera utilisée par le protocole de sécurité radio pour la protection des trames du WLAN.

Eléments de performances

Les empreintes MD5 et SHA1 découpent les données en blocs de 512 bits (64 octets). Le temps de calcul est en conséquence proportionnel à la taille du message, exprimée en nombre de blocs. Généralement les cartes à puce réalisent ces algorithmes par un code natif, les meilleures performances que nous avons mesurées sur des cartes JAVA (8 bits !) du marché sont de l'ordre de 11 ms par bloc (en moyenne), nous avons également remarqué que le temps calcul du SHA1 est environ deux fois supérieur à celui du MD5 (soit 7 ms pour MD5 et 15ms pour le SHA1)

Les calculs RSA sont très optimisés dans les cartes à puce, qui intègrent généralement un crypto processeur. Avec des tailles de clés inférieures à 2048 bits, on observe des temps calculs plus petits que 500ms.

Les procédures HMAC-MD5 et HMAC-SHA1 ne sont pas supportées par la norme javacard 2.2. Dans le contexte TLS, un calcul HMAC comporte deux appels à une fonction MD5 ou SHA1, avec une taille d'environ cinq blocs. Une instance HMAC coûte environ 10 blocs (soit 110 ms) plus un surcoût logiciel java (de l'ordre de 200 ms dans notre implémentation la plus performante)

Le protocole TLS échange beaucoup de données, de l'ordre de 100x64 octets (6400 octets), sur lesquelles sont réalisées divers calculs d'intégrité (MD5, SHA1). Les transferts d'information vers la carte à puce sont plutôt lent, de l'ordre de 1,5ms/octet, ce paramètre résulte de deux facteurs, d'une part le débit des entrées/sorties (9600 à 112,200 bits/s) et d'autre part les temps d'écriture dans la mémoire E^2PROM (de l'ordre de la ms par octet).

Enfin le protocole RC4, très populaire pour le chiffrement des tunnels SSL/TLS n'est pas supporté par JC2.2. Sa réalisation java coûte environ 1.6s (initialisation et chiffrement de 32 octets) dans notre implémentation.

Quelques formules magiques

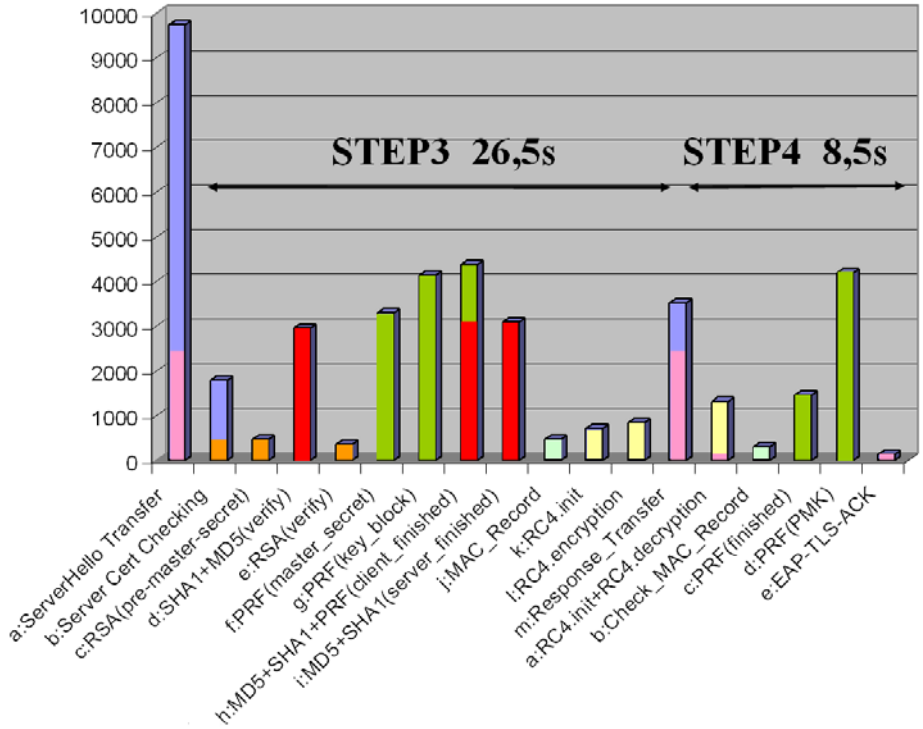

Figure 6: Exemple de répartition des temps calculs dans une carte EAP-TLS

Coût cryptographique global du protocole EAP-TLS.

(I) $\text{Transfer}_{100\text{blocs}}$ + 3x RSA + 66x $\text{HMAC}_{10\text{blocs}}$ + 6x $\text{HASH}_{100\text{blocs}}$ + 2x $\text{RC4}_{32\text{bits}}$

Soit 9,6 + 1,5 + 19,8 + 6,6 + 3,2 = **34,1 s**

Coût de l'étape 3, la plus critique en termes de temps de réponse

(II) $\text{Transfer}_{3\text{Ko}}$ + 3x RSA + 40x $\text{HMAC}_{10\text{blocs}}$ + 6x $\text{HASH}_{100\text{blocs}}$ + 1x $\text{RC4}_{32\text{bits}}$

Soit 4,5 + 1,5 + 12,0 + 6,6 + 1,6 = **26,2s**

En négligeant le temps calcul RC4 et en supposant un futur support des fonctions HMAC par les cartes java, le coût global devient

(III) $\text{Transfer}_{100\text{blocs}}$ + 3x RSA + 1260x blocs

Soit 9,6 + 1,5 + 13,9 = **25 s**

Ces éléments montrent que certaines cartes java du marché sont capables de réaliser l'étape 3 du protocole TLS en moins de 30 secondes, ce qui correspond au délai maximal de réponse à une requête EAP (par défaut), spécifiée par la norme IEEE 802.1x. Cependant, de réelles améliorations de performances ne seront possibles que part l'intégration des fonctions de HASH aux crypto processeurs. Le temps de transfert important observé entre carte et terminal ne pourra être réduit que par l'introduction de mémoires de type FLASH ou l'augmentation des tailles de RAM, dont les temps d'écritures sont peu importants.

Le mode Session Resume

Pour des raisons économiques (absence de crypto processeur RSA) ou de faisabilité (les environnements PKI sont réputés difficiles à gérer), certaines organisations, disposant par exemple d'un environnement centralisé, peuvent choisir de déployer une authentification à base de PSK (secrets partagés). Le mode de reprise de session de TLS répond à ces critères. Ainsi que nous l'avons décrit dans [6], il est possible d'émettre des cartes EAP-TLS, personnalisées avec un ou plusieurs *MasterSecret*, identifiés par des *Session-ID*s.

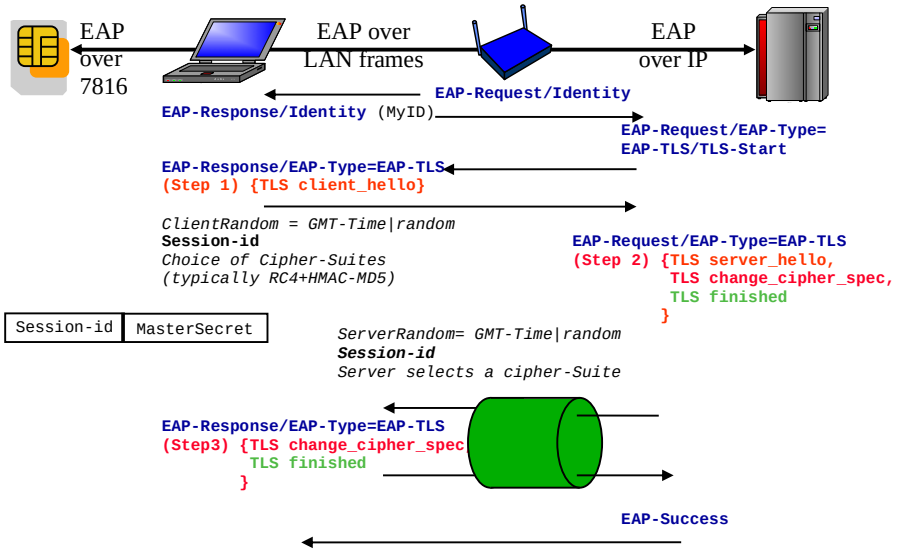

Figure 7: Session Resume de TLS

Une reprise de session se déroule en 3 phases.

- Phase 1. Le serveur démarre la session grâce à la requête EAP-TLS/start. Le client répond à cette dernière par le message TLS, *Client-Hello* qui contient un *Session-ID*.

- Phase 2. Le serveur retrouve le *MasterSecret* associé au *Session-ID* reçu. Il calcule de nouvelles clés de session à l'aide des nombres aléatoires générés par les deux parties. Il répond par le message TLS *Server-Hello* et prouve sa connaissance du *MasterSecret* en utilisant le tunnel TLS.

- Phase 3. Le client vérifie la réponse du serveur et prouve à son tour sa connaissance du *MasterSecret* en utilisant le tunnel TLS. Le client calcule la clé PMK utilisée par le protocole de sécurité radio.

Une formule magique pour le mode Session Resume

L'absence de certificats et de données chiffrées par des clés RSA réduit considérablement le volume des informations échangées, moins de 200 octets dans un cas concret. Le poids des fonctions de *hash* est donc moindre relativement aux performances du mode *Session Resume*. Le nombre d'appels aux méthodes HMAC est également réduit, car la fonction PRF est invoquée seulement quatre fois dans ce cas (au lieu de cinq appels dans le mode TLS standard).

Le *coût cryptographique de l 'étape 3* est le suivant,

$$\text{(IV) } \text{Transfer}_{200\text{octets}} + 50\text{x } \text{HMAC}_{10\text{blocs}} + 4\ \text{hash}_{3\text{blocs}} + 2\text{x RC4}$$

$$\text{Soit } 0{,}3 + 15{,}0 + 0{,}13 + 2{,}6 = \mathbf{18{,}0s}$$

En négligeant le délai de transfert des données et les temps de calcul RC4, le coût cryptographique de la reprise de session s'écrit

$$\text{(V) } 512\text{x blocs}$$

Ce mode est donc au moins de fois et demi plus rapide que le mode standard de TLS (1260x blocs)

Conclusion

La naissance des cartes EAP-TLS est un point de rupture dans l'usage des cartes à puce, et marque la disponibilité de véritables PKI de poche. Bien qu'il soit nécessaire d'améliorer les performances, tout en conservant un très faible prix, cette technologie émergente est une solution crédible pour le

déploiement d'une informatique enfouie sécurisée. Elle marque le premier jalon de l'introduction des puces sécurisées dans l'IP sans fil.

Bibliographie

[1] RFC 2246 "The TLS Protocol Version 1.0", 1999

[2] RFC 2716 " PPP EAP TLS Authentication Protocol ", 1999

[3] RFC 3748 "Extensible Authentication Protocol (EAP) ", 2004

[4] Internet Draft, "draft-urien-eap-smartcard-06.txt", 2004

[5] Mohamad Badra, Pascal Urien. "Toward SSL Integration in SIM SmartCards" IEEE Wireless Communications and Networking Conference 2004". Atlanta (Georgia, USA), March 2004.

[6] Pascal Urien, Mohamad Badra, Mesmin Dandjinou, "EAP smartcards, from dream to reality", 4th Workshop on Applications and Services in Wireless Networks, Boston, Massachusetts, USA, August 8-11 2004.

[7] G.Pujolle, M.Loutrel, P.Urien, P.Borras et D.Plateau, "Sécurité Wi-Fi", Eyrolles 2004