Espace Solutions

Exhibitors' conferences

Les conférences des exposants (Espace Solutions) se déroulent dans 2 salles situées à proximité immédiate des stands. Ces espaces sont gratuitement ouverts au public professionnel. Ils sont constitués de sessions de présentation et de tables rondes dans lesquelles s'expriment les exposants d'INFOSEC 2000 pour débattre des solutions concrètes, opérationnelles et effectivement commercialisées.

SALLE BLEUE

SALLE ROUGE

MARDI 6 JUIN 2000

11 H 00 : La sécurité sur Internet - **SYMANTEC**

12 H 00 : La réponse TREND MICRO à l'évolution virale et aux nouvelles menaces - **TREND MICRO FRANCE**

14 H 00 : Protection physique des systèmes d'information de l'entreprise - **NOBLET EQUIPEMENT**

15 H 00 : Sécurité des sites Internet : le test à distance fait son apparition - **CYRANO France**

16 H 00 : De l'empreinte digitale à la signature électronique : la sécurité informatique du 21e siècle - **ZALIX BIOMETRIE**

11 H 00 : MYCEA™ : les services INTEGRALIS de management et de surveillance de votre sécurité - **INTEGRALIS**

12 H 00 : Internet token : solution sécurisée de commerce électronique - **RAINBOW TECHNOLOGIES**

13 H 00 : Application Security Middleware : une plate-forme stratégique pour sécuriser votre environnement E-Business - **UBIZEN**

14 H 00 : Sécurité IP : clé de voute des réseaux d'entreprise - VPN - **CHECKPOINT SOFTWARE TECHNOLOGIES**

15 H 00 : Gestion des plans de crise et des plans de secours avec le logiciel XT-BORA - **EXPLOITIQUE**

16 H 00 : eToken et les architectures PKI - **ALADDIN KNOWLEDGE SYSTEMS**

MERCREDI 7 JUIN 2000

10 H 00 : La délinquance informatique - **ARES / LEXSI**

11 H 00 : Real-time security awareness throughout the enterprise - **ALLEN SYSTEMS GROUP**

12 H 00 : L'audit de sécurité : pourquoi et comment ? - **ATTEL**

10 H 00 : Fundamental E-Business Security - **ARGUS SYSTEMS GROUP (EUROPE)**

11 H 00 : NetWall 5.2 : une nouvelle approche de l'administration des firewall - **BULL - Div.BULLSOFT**

12 H 00 : Security as a service - **F-**

14 H 00 : La garantie de la confidentialité complète de l'information au sein des entreprises e-Business, avec CENTURA SQLBase SafeGarde - **CENTURA SOFTWARE**

15 H 00 : Tiers de confiance postal par Internet - **POSTEASY**

16 H 00 : Solution globale de sécurité logique - **EXPLOITIQUE**

SECURE

14 H 00 : PKI : anytime, anywhere, anydevice - **ENTRUST TECHNOLOGIES**

15 H 00 : La solution ultime aux pertes de données : récupération de données et recherche de preuves informatiques - **PCM ASSISTANCE**

16 H 00 : Gestion et protection du parc informatique et bureautique : une solution novatrice "Keylan" - **TYCO Electronics / AMP**

JEUDI 8 JUIN 2000

10 H 00 : Contraintes de l'utilisation de la cryptographie en entreprise et solutions - **AB SOFT**

11 H 00 : Sécurité IP distribuée : les accès réseau au coeur de la problématique sécuritaire - **SOLSOFT**

12 H 00 : La technologie ASQ (Active Security Qualification) ou l'intelligence de la sécurité - **NETSAQ**

14 H 00 : L'analyse de contenu au service des transactions sécurisées sur Internet - **TREND MICRO FRANCE**

15 H 00 : De l'intérêt du chiffrement matériel par rapport au chiffrement logiciel - **ATTEL**

10 H 00 : Nouveaux enjeux de la sécurité : sécurisez vos applications réseaux, Intranet / Extranet, avec une architecture d'autorisation - **IMECOM / TIVOLI**

11 H 00 : Sécurisation d'un site portail e-commerce - **ABAX PARTNER**

12 H 00 : Comment associer analyse de contenu, chiffrement et signature électronique : la solution Secretssweeper - **CONTENT TECHNOLOGIES**

14 H 00 : PKI, la dynamique de votre sécurité et de votre développement e-commerce - **UTIMACO SAFEWARE FRANCA**

15 H 00 : NetSecure Log : la solution intelligente d'analyse et de traitement des journaux d'événements des équipements de sécurité - **NETSECURE SOFTWARE**

"La carte à puce internet, une architecture ouverte et adaptée aux applications distribuées multimédia sécurisées"

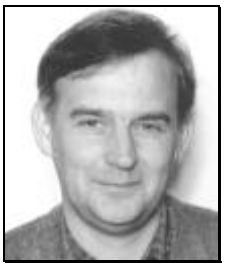
Pascal Urien
Ingénieur de Recherches Bull CP8.
Bull CP8, 68 route de Versailles BP 45, 78431 Louveciennes Cedex
eMail: Pascal.Urien@Bull.net
Tel: 01 39 66 42 30
Fax: 01 39 66 44 02

Hayder Saleh
Doctorant Cifre

Adel Tizraoui
Doctorant Cifre

Bibliographie.

Pascal Urien



Pascal Urien est ingénieur de l'école Centrale de Lyon et Docteur en Informatiques. Il a déposé une dizaine de brevets dans des domaines relatifs aux réseaux et aux cartes à puces. Il est responsable d'un programme de recherches (Bull CP8) sur l'intégration des puces électroniques dans les réseaux et architectures distribuées sécurisées, et des applications de ces technologies au commerce électronique. Il enseigne également les technologies IP dans différentes universités.

Hayder Saleh

Hayder Saleh est titulaire d'un DEA MISI et réalise une thèse cifre sur la définition d'architecture de communication sécurisé.

Adel Tizraoui

Adel Tizaoui est ingénieur INI.et titulaire d'un DEA MISI Il poursuit une thèse cifre relative à la sécurité du commerce des objets virtuel dans l'internet.

Résumé

Cette communication présente les grandes lignes d'une nouvelle technologie de carte à puce internet (baptisée *Oversoft*). Une puce internet partage la pile TCP/IP du terminal auquel elle est connectée, et se comporte comme un véritable nœud du réseau internet. En particulier elle est capable d'intégrer des applications client ou serveur du monde internet, c'est à dire définies par des RFCs (Request For Comment). Nous avons réalisé une première génération de cartes, à base de technologie Java, qui intègre un serveur web et un *trusted* proxy et qui se connecte de manière autonome au réseau. Nous décrivons l'application de ce concept à quelques services présents sur le web, et nous introduisons la notion d'objets distribués éventuellement multimédia, nécessitant des mécanismes de sécurité tels que authentification ou confidentialité

Summary

This paper presents the main features of a new technology, called Oversoft, which defines an innovative concept of internet smart Card. An internet card shares the TCP/IP stack of the terminal to which it is connected, and acts like a true internet node. It is able to support internet server or client applications, which are defined by RFCs (Request For Comment) standards. Our first internet card, which uses the JavaCard technology, includes a web server and a *trusted* proxy, and performs autonomous connection to internet network. We describe how this technology can work with today web services, and we introduce the notion of virtual objects, which require security functions.

La carte à puce internet, une architecture ouverte et adaptée aux applications distribuées multimédia sécurisées.

Pascal Urien - Hayder Saleh - Adel Tizraoui
Bull CP8, 68 route de Versailles BP 45, 78431 Louveciennes Cedex
eMail: Pascal.Urien@Bull.net

1. Introduction

Nous avons introduit une nouvelle architecture permettant de transformer une carte à puce en un nœud (client / serveur) du réseau internet. Dans ce contexte un navigateur réalise naturellement l'interface utilisateur associée à une carte, d'une part cette dernière se comporte comme un serveur web usuel, ce qui rend intuitif sa mise en œuvre, d'autre part la puce peut exécuter toute application internet (c'est à dire définie par un Request For Comment - RFC), c'est à dire accessible à partir du protocole TCP/IP.

2. La carte à puce traditionnelle

La carte à puce à microprocesseur (*SPOM* - **S**elf **P**rogrammable **O**ne-chip **M**icrocomputer) a été inventée au début des années 80 par Michel Ugon [6][7][8]. A travers le GIE cartes bancaires des dizaines de millions d'exemplaires ont été mis en circulation [9]. La carte à puce classique c'est une tranche de silicium de 25 mm^2, qui comporte un microprocesseur 8 bits (6805 ou 8051) cadencé à 3.3 Mhz (avec une puissance de l'ordre du MIPS) , une RAM de quelques centaines d'octets, des mémoires non volatiles ROM et EEPROM de quelques dizaines de Kilo-octet. La puce réalise schématiquement deux types de services :

- Des algorithmes cryptographiques utilisés pour le chiffrement, l'authentification, la génération de certificat.
- Le stockage de données qui sont protégées en lecture ou écriture par différentes méthodes d'authentification.

La qualité principale d'une carte à puce est d'être *Tamper Resistant*, c'est à dire qu'elle résiste à des attaques extérieures menées dans le but de lire les données ou le code stocké dans la puce. La plupart des attaques contre ces dispositifs sont dites de classe 3, c'est à dire qu'elles doivent être conduites par des organisations disposant de fonds importants et capables de supporter des recherches longues et coûteuses. Dans la plupart des cas il existe des méthodes plus commodes pour obtenir l'information secrète logée dans une puce. Les attaques contre les cartes peuvent être classées en trois différentes catégories, les attaques contre le micro-contrôleur (exploitation des défauts hardware), les attaques physiques (reverse engineering), et les attaques à bases d'analyses sophistiquées (par exemple Differential Power Analysis DPA). La sécurité d'une carte à puce est un équilibre entre la sécurité physique associée à la puce et le système d'exploitation. Ce dernier est organisé autour d'une interface de communication série. Il nécessite des capacités mémoire de l'ordre de quelques dizaines de Kilo-octets, supervise la sécurité du micro-contrôleur, assure la protection des données (grâce à des méthodes d'authentification) et réalise divers algorithmes cryptographiques (DES, RSA ...).

Un inconvénient d'un tel objet informatique est de ne posséder aucun périphérique d'entrée sortie classique tels que, un clavier, un écran ou une carte de communication. Il est donc nécessaire d'associer à une carte d'une part un lecteur (qui réalise des services de bas nivaux tels que alimentation en énergie et en horloge), et d'autre part un terminal qui va offrir des ressources d'entrées sorties et par exemple un accès réseau.

3. Les architectures à base de carte à puce.

Les architectures à bases de cartes se sont historiquement développées dans un contexte bancaire, le terminal est réputé sûre, le couple terminal carte est identifié de manière visuelle, ainsi une carte bleue fonctionne avec un terminal carte bleue.
La norme ISO7816 [1] définit le format des ordres qui sont échangés entre une carte et le terminal, que l'on nomme encore dans l'appendice iso 7816-4, *APDU* (Application Protocol Data Unit) . Deux types de protocoles de transmission ont été normalisés, *T=0* qui est orienté caractère (échange d'octets munis d'un bit de parité) et *T=1* qui utilise un mode bloc (transfert d'un ensemble d'octets munis d'une somme de contrôle).
Le terminal émet une commande (.command), la carte répond (.response). De fait l'ISO a normalisé un format de message, mais pas leur sémantique, c'est à dire qu'il n'existe aucun moyen pour envoyer à priori les *bonnes* commandes à la *bonne* carte. De surcroît aucun procédé normalisé ne réalise l'identification une carte particulière, c'est donc une information connue par ailleurs qui permet de constituer un couple correct carte terminal.

4. Les architectures ouvertes Java, SCW.

L'industrie de la carte à puce subit une mutation, de nouveaux usages voient le jour (cartes GSM SIM, cartes santé, cartes RSA, cartes de transport [2][3][4]), le taux de croissance de ce secteur est important (on prévoit un taux de l'ordre de 40 %), l'appât des volumes (1 milliard de cartes seront vendues courant 2000) attire de nouveaux acteurs (Microsoft ...).
Les cartes traditionnelles sont mono - application, elles sont en fait conçues et fabriquées pour une fonction unique liée à un opérateur particulier (banques, opérateur de téléphonie mobile, ministère de la santé ...); de manière générale le porteur ne contrôle pas l'information qu'il transporte.
En 1997 est apparu la JavaCard [11] qui intègre une machine virtuelle Java, c'est le premier pas vers des architectures multi applications écrites bien évidemment en Java. Ce langage permet de réaliser des services qui autrefois auraient impliqués la conception (coûteuse) d'une carte particulière. Les applets logés dans les cartes peuvent implémenter des fonctions telles que porte monnaie électronique, programme de fidélisation ....
Cependant l'introduction du langage Java rend encore plus complexe l'usage des cartes, en effet la multiplication des services disponibles implique un nombre encore plus important de logiciels spécifiques (nommés *Application Protocol* dans la terminologie OCF [5]) qui doivent être présent sur chaque terminal ou la carte sera utilisée.
En 1999 Microsoft [12] a annoncé Smart Card for Windows (SCW), une carte à puce qui sera disponible avec son prochain système d'exploitation, courant 2000. La compagnie justifie son choix par la conviction que les cartes sont une composante clé de la sécurité (et donc du commerce électronique), et que pour compenser l'absence d'un standard de haut niveau pour leur mise en œuvre, une des solutions est d'offrir une interface familière à un utilisateur, par exemple le *look and feel* windows. La carte (qui se programme en basic) est donc organisée autour d'un système d'exploitation largement diffusé, qui permet à son porteur de la mettre en œuvre facilement (à l'image par exemple d'une disquette, ou d'une machine du *voisinage*

réseau). En conclusion cette approche consiste à associer à chaque objet présent dans la carte une API de niveau Système d'Exploitation.
Cette architecture a l'inconvénient de n'être réellement ouverte pour un système d'exploitation unique, hors tous les experts s'accordent à penser que seulement 40 % des terminaux qui accéderont à internet seront des ordinateurs personnels en 2001.
Notre modèle s'applique à utilisateur mobile qui utilise des terminaux variés (*ubiquitous computing* [10]) accèdent à internet (*ubiquitous internet*), citons par exemple

- Les ordinateurs personnels.
- Les téléphones mobiles (WAP et GPRS permettront un accès à internet courant 2000).
- Les assistants personnels(les Palm Pilot VII sont reliés par internet par une technologie sans fil sur le territoire américain).
- Les consoles de jeu (la console dreamcast comporte un modem intégré, par exemple pour jouer en réseau)
- Les lave-linge (le groupe Merloni a récemment annoncé 12/99 une technologie domotique et prône des services du type *Pay Per Wash*)

Ces terminaux banalisés permettront d'accéder à des services divers pour lesquels des mécanismes d'authentification de l'utilisateur ou de paiement seront nécessaires, et pour lesquels la carte à puce est un vecteur idéal. Nous pensons donc que la mise en œuvre universelle d'une carte (c'est à dire sans avoir à imposer à l'utilisateur un système d'exploitation ou une configuration particulière) est la clé du succès des futurs services qui seront créés au moyen du réseau internet.

5. La carte à puce internet.

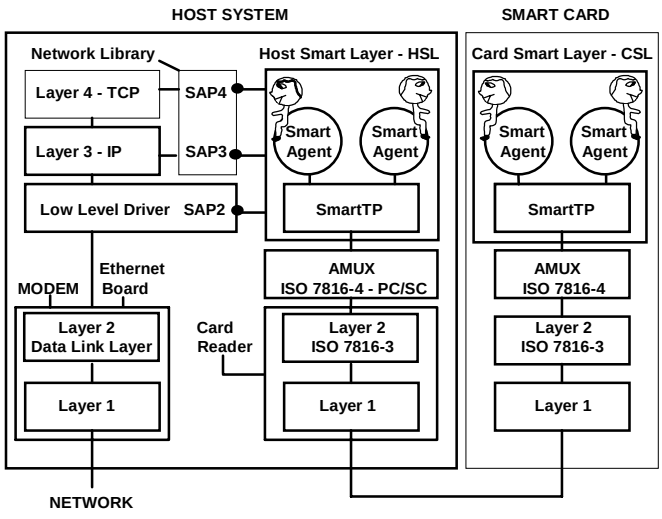


Figure 1 - Carte à puce Internet

La carte à puce internet (Cf figure 1) [13], inventée par Bull CP8 en 1998 [14], encore baptisée technologie *Oversoft* a pour objectif de transformer cette dernière (associée à un terminal) en un nœud internet. En conséquence la puce se comporte comme un serveur et/ou un client TCP/IP.
L'intérêt primordial d'un serveur web logé dans une carte est de permettre à un navigateur web de communiquer avec la puce. Par exemple l'url http://127.0.0.1:8080 donne accès à la page html de garde (home page) de la carte qui dresse une liste (associée à des hyperliens) des ressources hébergées. Les ressources sont des objets (éventuellement multimédia) tels que page html, images, applets ou connexion à un serveur distant. Certaines d'entre elles peuvent être associées à un *pin code* (l'équivalent du code d'une carte bleue ou d'une carte SIM) qui sera demandé à l'aide d'un formulaire html.
L'association simultanée d'un serveur et d'un client localisés dans une carte permet d'introduire la notion de *trusted proxy*. Un tel objet associe à une url carte (par exemple http://127.0.0.1:8080/eMail) une procédure de connexion avec un serveur web externe (par

exemple un serveur de courrier gratuit). On introduit ainsi la notion de *fichier virtuel*, c'est à dire un objet qui est associé à une url carte, mais qui en fait est logé sur un serveur distant, quelque part dans le web. La carte se comporte comme une interface d'accès à cet objet, elle sera par exemple responsable de la procédure d'authentification.
La carte à puce internet est *url centric*, toutes les ressources qui lui sont associées sont identifiées par une url spécifique.

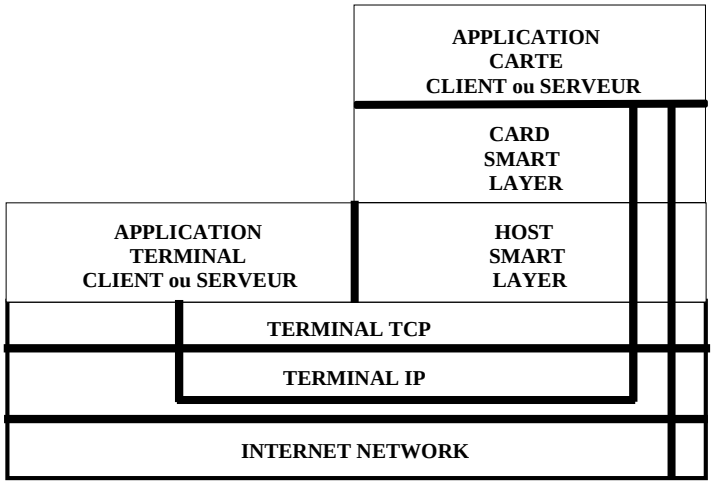


Figure 2 Architecture logique d'une carte à puce internet

La carte à puce internet partage en fait la couche TCP/IP qui est présente sur le terminal auquel elle est reliée, en particulier elle hérite de l'adresse IP du terminal, et de toute sa configuration réseau. Pour réaliser cet objectif nous avons défini une nouvelle pile protocolaire dénommée Smart Layer, qui est organisée autour d'un protocole original SmartTP (Smart Transfer Protocol), une version allégée du protocole TCP.
Une application internet (client ou serveur) logée dans la carte s'appuie sur une pile Card Smart Layer, qui communique avec une entité analogue située dans le terminal (Host Smart Layer) Cette dernière possède une interface d'accès avec la pile TCP/IP du terminal, d'un point de vue logique (Cf figure 2) une application résidente dans la carte a donc accès à la pile TCP/IP du terminal, de manière tout à fait analogue à une application terminal usuelle. Grâce au mécanisme d'adresse ip de boucle (*loopback address* 127.0.0.1), une application internet (terminal), peut également échanger des données via la pile TCP/IP avec une application résidant dans la carte.

6. Architecture.

SAP

Dans l'architecture TCP/IP une application réseau (telle que un navigateur web par exemple) s'appuie sur un modèle qui comporte les 4 premières couches du modèle OSI Les couches 1 et 2 représentent par exemple la carte (physique) ethernet ou un modem. La machine voit (utilise) une ressource réseau (carte ...) à travers une couche d'interface logicielle particulière encore nommée driver couches basses (ainsi NDIS est une interface couche basse développée par Microsoft). De fait, utiliser une carte réseau dans un système c'est mettre en œuvre un driver couches basses, en terminologie ISO cette interface peut être vue comme un ***SAP*** (Service Access Point) de niveau 2 (SAP2). De façon analogue une application utilise les couches TCP/IP au moyen d'une bibliothèque réseau fournie par le système hôte, une application réseau sait mettre en œuvre une telle bibliothèque à travers des SAP de niveaux 3 (SAP3) ou 4 (SAP4). Par exemple winsock.dll est la bibliothèque dynamique qui permet d'utiliser TCP/IP sur une machine windows.

Smart Layer

L'architecture comporte deux piles protocolaires (Smart Layer) l'une logée sur le terminal (Host Smart Layer HSL) et l'autre dans la puce (Card Smart Layer - CSL). Chaque pile s'appuie sur des couches OSI 1 et 2, définies en fait par les normes ISO 7816 (1,2,3).

Amux

Côté carte une entité parfois appelée APDU Manager analyse le flux des APDUs entrantes et réalisent les opérations nécessaires. Par exemple dans une carte multi applications, une APDU *SELECT* est utilisée pour sélectionner une application particulière (un programme écrit en java, ou encore un cardlet). Cette opération étant réalisée, toutes les APDUs reçues sont routées vers l'application en cours. Dans un terminal un logiciel particulier gère le lecteur de cartes. En 1996 un ensemble d'industriels de la carte à puce et de systèmes informatiques ont adopté le standard PC/SC qui permet l'intégration de lecteurs et de cartes dans les machines informatiques. Une application émet/reçoit des APDUs vers/depuis un lecteur à l'aide d'APIs (Application Programmatic Interface) délivrées par le système d'exploitation. Nous appellerons AMUX (Apdu MUltipleXeur) la couche logicielle logée sur le terminal ou la carte, qui réalise la commutation des APDUs vers les applications utilisatrices.

SmartTP

Nous avons défini un protocole inspiré de TCP (Smart Transfer Protocol - SmartTP) qui est transporté par des APDUs spécifiques, c'est à dire qu'une entité SmartTP émet et reçoit des APDUs vers/depuis une couche AMUX. Une unité de donnée SmartTP (SmartTP pdu, *protocol data unit*) comporte une référence de destination (d), une référence source (s), un champ *flag* de 8 bits (Open-Read-Write-Close), et de manière optionnelle un suite d'octets de donnée (*data*). Nous appelerons *token*, un SmartTP pdu qui ne transporte aucune donnée. Une référence est l'équivalent d'un port TCP ou UDP. L'entité SmartTP se comporte comme un commutateur qui dirige les SmartTP pdu depuis/vers *des entités logicielles autonomes* nommées Agents.

Agents

Deux Agents sont mis en relation par une session. Un flag OPEN définit l'ouverture d'une session, un flag CLOSE indique la fermeture d'une session. Un agent peut se bloquer c'est à dire émettre un pdu avec un flag BLOCK positionné, et suspendre toute activité jusqu'à la réception d'un nouveau pdu. Deux flags additionnels (READ et WRITE) permettent de construire le paradigme OPEN-READ-WRITE-CLOSE (bloquant ou non) utilisé dans les systèmes Unix pour la manipulation des fichiers.
Une session est identifiée par un couple de références. Un agent client émet un SmartTP pdu dont l'indicateur Open est positionné. La référence d'un client est éphémère, elle est allouée par l'entité SmartTP au moyen d'une primitive GiveReference(), par exemple à l'aide d'un algorithme du type LRU (Last Recently Used). Un agent serveur reçoit un SmartTP pdu avec l'indicateur OPEN positionné. La référence d'un serveur est fixe, et connue par convention (well known value). Une session est identifiée par le couple des références client et serveur dont la valeur est pseudo unique (c'est à dire unique durant un intervalle de temps suffisant). Un SmartTP pdu dont l'indicateur Close est positionné est la marque d'une fin de session. Un client notifie à l'entité SmartTP la fin d'une session au moyen d'une primitive ReleaseReference() qui entraîne l'abandon de la référence éphémère.

Côté terminal un agent réseau peut accéder à un SAP (par un exemple un SAP de niveau 4, c'est à un point d'accès sur la couche TCP ou UDP du terminal), il se comporte en fait comme un traducteur de protocole TCP/SmartTP ou UDP/SmartTP.

Http

Une session entre un agent carte web (qui implémente le protocole http, rfc 2068) et un agent terminal réseau (par exemple un serveur TCP sur le port 8080) permet de transformer une carte à puce en un serveur web qui utilise la pile TCP/IP du terminal auquel la puce est reliée mais qui réalise le protocole HTTP.

Trusted Proxy

Une application carte mise en œuvre à partir d'une URL (c'est une notion similaire au CGI) peut accéder au web à l'aide d'une session agent-carte / agent-réseau-client-TCP, l'association de deux sessions simultanées permet de réaliser un proxy qui relie par exemple un navigateur à un serveur externe. C'est la notion de fichier virtuel ou encore *trusted proxy* (Cf figure 3).

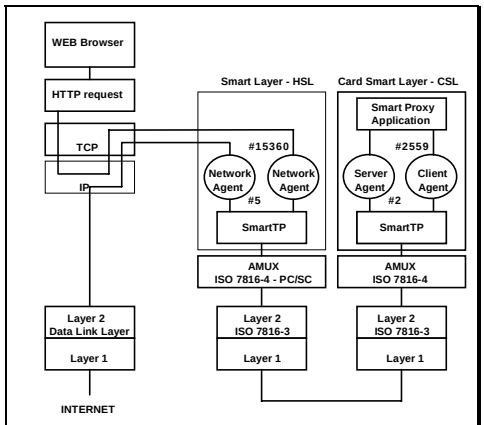


Figure 3 - Trusted Proxy

Fichier Virtuel

Un fichier virtuel associe à une url carte une procédure de connexion à un serveur externe. Les données reçues par la carte sont ensuite traitées et retransmises vers le navigateur. Voici par exemple un scénario possible:

1. Une connexion est établie avec un serveur (web) logé dans la carte. Cet agent (#2) échange des données à travers une session avec un agent réseau (#15360).
2. Des données sont échangées entre le client TCP (navigateur) et le serveur de la carte. A partir de ces informations l'application proxy est capable de déterminer l'adresse d'un serveur internet.
3. L'application carte ouvre une connexion avec un serveur internet. Une nouvelle session est crée entre un agent client carte et un agent réseau. L'agent client carte obtient une référence éphémère (#2559) et envoie un pdu *Open+Block* à un agent réseau (client TCP) dont la référence est prédéterminée (#5). Les données associées à ce pdu précisent l'adresse et le port du serveur distant. L'agent réseau (#5) ouvre une connexion TCP, et en cas de succès émet un jeton à destination de l'agent #2559.
4. L'agent carte #2559 échange des données avec le serveur internet distant, grâce à la session ouverte avec l'agent réseau #5.
5. Les données reçues depuis un des agents réseaux sont traitées et la cas échéant retransmises vers le deuxième agent réseau par le smart proxy.
6. Un des deux agents réseaux referme la session associée parce que la connexion TCP qu'il gère a pris fin.

La figure 4,illustre l'échange de données (SmartTP pdu) lors de l'accès à un fichier virtuel carte, depuis un navigateur.

*Une première session est ouverte.***T**[s=15360,d=2,open+block], **C**[s=2,d=15360].
Le navigateur web envoie une requête http au serveur web de la carte.
T[s=15360,d=2,write+block,data], **C**[s=2,d=15360,ack].
La dernière partie de la requête est émise. **T**[s=15360,d=2,write+block,data].
Le proxy carte ouvre une deuxième connexion avec un serveur internet.
C[s=2559,d=5,open+block,data], **T**[s=5,d=2559,block].
Le proxy carte émet une requête http vers le serveur internet.
C[s=2559,d=5,write+block,data], **T**[s=5,d=2559,block].
La dernière partie de la requête est émise.
C[s=2559,d=5,write,data].
Le proxy carte relaie les informations reçues par les agents réseaux.
T[s=5,d=2559,block], **C**[s=2,d=15360,block], **T**[s=15360,d=2], **C**[s=2559,d=5].
Des données sont reçues depuis internet.
T[s=5,d=2559,write+block,data], **C**[s=2,d=15360,write+block,data].
T[s=15360,d=2,block], **C**[s=2559,d=5,block], **T**[s=5,d=2559] **C**[s=2,d=15360].
L'agent #5 est à nouveau prêt à recevoir des données du réseau..
Fin de la session.
Le proxy carte retransmet des données
T[s=5,d=2599,write+block], **C**[s=2,d=15360,write+block].
L'agent #15360 émet un jeton (bloquant)
T[s=15360,d=2,block], **C**[s=2559,d=5,block].
Le navigateur met fin à sa connexion TCP, l'agent #15360 referme sa session.
T[s=15360,d=2,close], **C**[s=2559,d=5,close].
L'agent #5 ferme session avec l'agent #2559, mais deux pdu produits avant cet événement sont envoyés par HSL vers CSL et entraînent l'émission de deux jetons par CSL.
T[s=5,d=2559], **C**[s=0,d=0].**T**[s=5,d=2559,write+block,data], **C**[s=0,d=0].

Figure 4 - échange de SmartTP pdu, lors de l'accès à un fichier virtuel

7. Applications

Il est évident que l'organisation d'une puce autour d'un serveur web permet de révolutionner son usage, d'une part un protocole unique (SmartTP) permet d'accéder aux cartes, d'autre part tout utilisateur d'un navigateur (et ils sont nombreux!) connaît à priori le mode d'emploi associé. Nous allons à présent décrire quelques exemples d'intégration des cartes internet à des services web usuels.

HTTP 1.1 et authentification

La RFC 2068 décrit un mécanisme d'authentification, à base de challenge qui est en fait utilisé sous sa forme la plus simple. Après la réception d'une requête (http) du client le serveur répond par le message

HTTP/1.1 401 Authorization Required
WWW-Authenticate: auth-scheme realm="realm-value" [, auth-param="value"]

Une procédure d'authentification est identifiée par le nom *auth-scheme* (par exemple *Basic*), elle est associée à un sous ensemble (realm-value) des ressources disponibles sur le serveur, de manière optionnelle un paramètre auth-param= "value" est utilisé pour fournir par exemple un nombre aléatoire (random = "ABEF456789"). Le client re-formule alors sa requête en incluant la ligne :

Authorization: realm #auth-param

Ainsi lorsque l'on utilise le schéma d'authentication *Basic* l'internaute voit s'afficher une boîte de dialogue qui l'invite à fournir son login et son password. Le navigateur génère alors une nouvelle requête dont la valeur associée à *Authorization:* comporte le login et le password séparé par le caractère : et encodé en *Base64*. On remarque que dans cette méthode couramment utilisée, le login et le mot de passe sont transmis en clair.
Un mécanisme de fichier virtuel permet de mettre en œuvre une procédure de challenge classique (F(random)), au terme de laquelle le serveur pourrait générer une notification de re direction avec la présence d'un *cookie*, la procédure pourrait par exemple être la suivante

1. A partir de l'url
   http://carte:8080/auth-scheme?Key=Ki&Target=www.host.com/index.html,
   la puce conduit une procédure d'authentification (auth-scheme) utilisant la clé Ki, dans le but d'obtenir le fichier identifié par l'URI http://www.host.com/index.html.
1. Au terme de la procédure d'authentification le serveur transmet à la puce une notification de re - direction comportant par exemple un cookie, ce message est alors retransmis par la carte vers le navigateur.

Authentification

La plupart des nombreux services disponibles sur le web sont accessibles au moyen d'une page html de garde qui se présente typiquement sous la forme d'un formulaire dont les deux champs à renseigner sont nommés login et password.(Cf figure 5) Le contenu du formulaire (par exemple login=Smart&password=Card) est transmis le plus souvent en clair (cependant dans certain cas le protocole SSL est utilisé) dans un message du type http POST. Le mécanisme de fichier virtuel augmente la sécurité, car le mot de passe n'est pas tapé sur le clavier de l'ordinateur, par exemple

Figure 5 - Un formulaire d'authentification

1. L'url http://carte:8080/eMail est un fichier virtuel
1. La puce se connecte à un serveur de courrier et fournit le login et le mot de passe de l'internaute.

```
HTTP/1.1 302 Redirect
Location: http://AnOtherSite.com/index.html
```

Figure 6 - Message de redirection

Redirection

La re direction (figure 6) est un mécanisme qui permet de rediriger une requête http vers un autre site. Un fichier virtuel peut être associé à un message de redirection dans ce cas la carte joue le rôle d'un annuaire (bookmark) actif.

1. L'url http://carte:8080/Service est un fichier virtuel
2. Le fichier Service est un message de redirection.

SSL

SSL est un protocole largement utilisé pour le commerce électronique. Résumons rapidement les principales étapes d'une session SSL.

1- Le client (le navigateur) débute une session avec un message *ClientHello* qui comporte un *Session_ID* (identificateur de session) dont la valeur est égale à zéro pour une nouvelle session et non nulle pour la reprise d'une session préalablement ouverte. Il propose un choix d'options disponibles, nommé *Cipher_Suites* et qui comporte :

- une liste de mécanismes d'échange de clés, en règle générale à base de clés publique RSA. La législation américaine limite les clés de chiffrement RSA *export* à 512 bits.
- un ensemble d'algorithmes de chiffrement tels que RC2, RC4, DES. Les clés sont limitées généralement à 40 bits, bien que depuis peu certains éditeurs supportent des clés (RC4) de 128 bits.
- une liste de fonctions de digests (usuellement MD5)

Le message comporte également un nombre aléatoire *ClientHello.random*. utilisé pour le calcul des clés de session.

2- Le serveur répond par un ensemble de messages, qui typiquement comporte un nombre aléatoire (*ServerHello.random)*, le choix d'un *Cipher_Suite*, une liste de certificats des clés publiques du serveur, et de manière optionnelle (et rarement utilisée) une demande d'authentification du client.

3- Le client analyse la validité du certificat du serveur, génère un *master_secret* (un mot de 48 octets) qui est chiffré avec la clé publique du serveur ($master\_secret^{ServerPublicKey}$). De manière optionnelle le client fournit un certificat, à l'aide duquel il réalise la signature du *master_secret*. Les clés de chiffrement et divers paramètres (IV, MAC Secret...) sont déduits du *master_secret* et des nombres aléatoires,

Key(s) = F(master_secret, ClientHello.random, ServerHello.random).

4- Le serveur calcule les clés et divers paramètres, la session SSL assure dès lors la confidentialité et l'intégrité des données échangées.

5- Lorsque le Session_ID est non nul, les deux extrémités calculent de nouvelles clés de session à partir des paramètres ClientHello.random, ServerHello.random, L'intégrité des messages est garantie par la conservation des valeurs courantes des fonctions de Hash.

- Le premier point faible du protocole SSL est la liste des possibles algorithmes de chiffrement proposés par le navigateur. Par exemple l'algorithme de chiffrement RC4 (dont la largeur de clé peut atteindre 128 bits) utilise couramment un clé de 40 bits. A raison de 1 million de clés par seconde il faut 12.7 jours pour générer toutes les clés. En juillet 1995, Damien Doligez, doctorant à l'INRIA a cassé un clé RC4 de 40 bits en une semaine, à l'aide d'une centaine de machines générant chacune de l'ordre de 10000 clés/seconde. Un pentium 2, 233 Mhz est capable de produire 64000 clés par secondes, une dizaine de PCs suffisent aujourd'hui pour le test d'un million de clés par secondes. Remarquons également que certain navigateurs supportent l'absence de chiffrement (*NULL Cipher*) tolérée par le protocole SSL.

- Le deuxième point faible est que le serveur choisit l'algorithme de chiffrement. Par exemple si un choix est possible entre une clé de 128 bits et 40 bits, le serveur peut opter pour la deuxième solution.

- Un autre point délicat est la vérification du certificat du serveur, qui est un point clé de la sécurité. Il est en effet nécessaire de faire confiance au code du navigateur qui analyse le certificat.

- Enfin la taille de la clé publique proposée par le serveur pour le chiffrement est également un point sensible. Pour mémoire, en Août 1999 le laboratoire hollandais CWI a réalisé la factorisation d'une clé RSA 512 bits à l'aide d'une puissance de calcul de l'ordre de 8400 MIPS-an, fournie par 300 ordinateurs et stations de travail en 7 mois.

Un proxy SSL logé dans une puce permet d'éviter ces inconvénients, si l'on admet que l'émetteur de la carte est digne de confiance. Ainsi une url (carte) http://127.0.0.1:8080/?ssl=www.bank.fr/urienp permet d'ouvrir au moyen d'un *trusted proxy* une session SSL. Dès lors la session SSL est entièrement gérée par la carte, qui en particulier réalise la vérification du certificat, et peut authentifier le client à l'aide d'un certificat logé dans la carte.

Un autre aspect important est le transfert d'une session SSL ouverte par une carte internet vers un terminal. Prenons par exemple un cas usuel d'application, la consultation d'un compte bancaire. Une session SSL est ouverte, l'internaute (c'est à dire la puce) fournit le login (le numéro de compte) et le mot de passe. Dès lors il est parfaitement possible de transférer au terminal le contexte de la session c'est à dire le type de fonctions (chiffrement et hash) préalablement choisies et les valeurs courantes des fonctions de hash. De nouvelles clés seront déterminées (via un nouveau *ClientHello*) par le client pour chaque nouvelle session http.

8. Objet Virtuel

Figure 7 - Notion d'objet virtuel

Nous pensons que notre architecture, couplée aux évolutions technologiques des puces (processeurs RISC 32 bits avec une puissance de l'ordre de 50 MIPS [15], évolution des mémoires EEPROM vers des capacités voisines du Mo) sera un moteur du développement du commerce de l'objet virtuel.
L'objet virtuel est un objet numérique que l'on peut utiliser sans support physique au moyen du réseau. Un tel objet est associé à des *méthodes d'accès* plus ou moins sécurisées et soumis à des *contraintes de débit*. Une négociation avec le réseau peut être nécessaire pour obtenir une qualité de service (QoS) particulière.
On peut considérer que l'accès à une messagerie électronique est un objet virtuel, un logiciel capable de lire ou de visualiser le courrier est une méthode associée à cet objet; la sécurité est réalisée par un login et un mot de passe.
De même le contenu d'un album cédérom est un objet virtuel. La technologie de compression au format MP3 permet d'accéder à cet objet avec des débits de l'ordre de 0.5 à 1Mo/minute compatibles avec les capacités du réseau internet. Notre vision du lecteur cédérom est en fait un terminal banalisé connecté à internet. Les enregistrements sont logés sur un serveur. L'utilisateur acquiert un certificat de propriété (définitive ou temporaire) qui est stocké sur sa carte à puce, ou sur un serveur distant. L'objet qu'il possède (CD) n'a pas de support

physique, il s'agit bien d'un objet virtuel. Un des avantages pour le propriétaire est la composition de programmes de son choix à partir d'un terminal quelconque et d'une carte à puce. De fait de nombreux objets virtuels font leur apparition sur internet, radio internet, journaux électroniques, télévision web. La carte à puce internet contrôle les accès à ces nouveaux services lorsqu'ils ne sont pas gratuits. Eventuellement cette opération constitue une méthode de rémunération en soit, par exemple la carte certifie que l'accès au service est réalisé grâce à un *internet service provider* (ISP) particulier.
En conclusion nous suggérons un modèle du commerce de l'objet virtuel, dans lequel un utilisateur mobile utilise des terminaux internet banalisés. La carte à puce internet réalise une interface d'accès à un serveur qui abrite un service particulier, lequel peut nécessiter un logiciel spécifique (parfois nommé *object viewer*, par exemple un abonnement à un journal implique l'utilisation d'un *reader* spécifique qui n'est pas forcément la propriété du journal), la facturation du service peut être réalisée au moyen d'un serveur de paiement. Enfin, si les capacités physiques de la carte sont trop exiguës, on pourra étendre son contenu à l'aide d'une base de données externe logée sur un serveur, la carte assurant dans ce cas la sécurité des communications nécessaires. On remarquera que la carte internet pouvant nativement accéder au web pour ses propres besoins, la mémoire virtuelle associée à ce dispositif est quasi illimitée.

9. Conclusion

Nous avons réalisé une première génération de carte internet. Les évolutions technologiques en terme de capacités mémoires et de puissances de traitement, permettent d'envisager l'utilisation d'une puce comme un véritable PC de poche, dont l'ergonomie est assurée par un navigateur.

10. Références

1. International Organization for Standardization, "Identification Cards - Integrated Circuit(s) Cards with Contacts", ISO 7816.
2. International Organization for Standardization, "Contactless integrated circuit(s) cards - Proximity Cards", ISO 14443.
3. European Telecommunications Standards Institute, "Digital cellular telecommunications system (Phase 2+) Specification of the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface" ETSI GSM 11.11
4. European Telecommunications Standards Institute, "Digital cellular telecommunications system (Phase 2+) Specification of the SIM Application Toolkit for the Subscriber Identity Module - Mobile Equipment (SIM -ME) interface", ETSI GSM 11.14.
5. OpenCard Framework - OCF - http://www.opencard.org.
6. "Le musée de la carte" - http://www.cardshow.com/museum/exposition.html
7. F.Guez C.Robbert A.Lauret - "Les cartes à microcircuit" - MASSON 1988
8. Louis Claude Guillou, Michel Ugon, and Jean-Jacques Quisquater, "The Smart Card: A Standardized Security Device Dedicated to Public Cryptology,", Contemporary Cryptology. The Science of Information Integrity, ed. Gustavus J.Simmons, IEEE Press 1992, pp. 561-613
9. Carol Hovenga Fancher, "In Your Pocket: Smartcards", IEEE Spectrum, February 1997, pp. 47 - 53.
10. Group D, I.S.206, SIMS - "Future Organisation of the Computer and Communications Industries" - http://www.sims.berkeley.edu/courses/is206/f97/GroupD/dfutre.html - 1997
11. Constantinos Markantonakis, "The Case for a Secure Multi-Application Smart Card Operating System", Information Security Workshop 97 (ISW'97), September 1997 (Ishikawa in Japan), Springer-Verlag (LNCS 1396), Berlin (1997), pp.188-197.
12. Microsoft - "SmartCard White paper" - http://www.microsoft.com/windowsce/smartcard - 1999
13. Pascal Urien, Hayder Saleh - "Une nouvelle approche de la carte à puce réseau" . Journées Nationales Réseaux JRES99 - Montpellier.
14. Pascal Urien - "Procédé de communication entre une station d'utilisateur et un réseau, notamment de type internet, et architecture de mise en œuvre" brevet déposé le 13 août 1988, N° d'enregistrement 98 10401, N° de publication 2 782 435
15. Jean Pierre Tual "MASSC, A Generic Architecture for Multiapplication Smart Cards", IEEE Micro September-October 1999.