Solution standard de compensation appliqué à une architecture e-Business sécurisée

Feltus, Christophe; Khadraoui, Djamel; Dulaunoy, Alexandre

Published in:
Proceedings of NOTERE 2004, Les Nouvelles TEchnnologies de la Répartition, Saida, Maroc.

Publication date:
2004

Document Version
Première version, également connu sous le nom de pré-print

Solution standard de compensation appliquée à une architecture e-Business sécurisée

Christophe FELTUS, Djamel KHADRAOUI
Centre d'Innovation par les Technologies de l'Information (CITI)
Centre de Recherche Public Henri Tudor
29, Avenue J. F.Kennedy L-1855 Luxembourg-Kirchberg
{nom.prénom}@tudor.lu **-** ***http://www.citi.tudor.lu***

Alexandre DULAUNOY
CSRRT-LU ASBL
10, rue du Faubourg
B-6811 Les Bulles (Chiny) - Belgium
adulau@foo.be

RESUME
L'article présente une expérimentation du protocole OpenSST[1] dans le cadre de compensation[2] appliquée à une architecture e-Business autour de laquelle a été greffé un composant chargé d'effectuer des demandes d'autorisation de paiement. Celle-ci est déployée sur une plate-forme e-Business[3]. OpenSST comme protocole pour le transport des messages d'autorisation de paiement procure certains avantages : sécurité, simplicité et ouverture. OpenSST est développé pour répondre à trois exigences majeures : atteinte d'un niveau de sécurité élevé, simplicité requise en ingénierie logicielle qui se révèle être un facteur important pour le design d'un système informatique, et ouverture du protocole à l'acceptation et à l'utilisation de standards existant, de façon à assurer un interfaçage aisé à d'autres solutions. Pour atteindre ces exigences, OpenSST a été bâti sur un format de message standardisé, ce qui se décompose d'une part par une structure de message, et d'autre part par le fonctionnement du protocole.

MOTS CLES : compensation, transactions e-business, WEB services, XML, sécurité informatique.

1. INTRODUCTION

Les transactions électroniques de payement connaissent une croissance continue depuis plusieurs années. Ce phénomène est principalement dû à la croissance du nombre d'utilisateurs connectés à l'Internet et au développement d'infrastructures d'accès à Internet à grande vitesse. Par ailleurs, si auparavant seules les moyennes et grandes entreprises pouvaient se permettre de pratiquer le commerce électronique en ligne, aujourd'hui, les petites entreprises profitent également de ce type de commerce. Etroitement liés au développement du e-commerce, les besoins en sécurité transactionnelle, comme la signature digitale côté client ou la non-répudiation apparaissent comme critiques pour le fonctionnement et la viabilité des transactions e-business. Les solutions de sécurisation constituent le fondement de toute activité e-business. Leur sélection est donc extrêmement sensible sur le plan de la qualité, de la sécurité et de la pérennité.

Les solutions et protocoles existants sont principalement de deux types: propriétaires ou non. Une société qui opte pour une solution propriétaire choisit d'acquérir la solution à un prix plus ou moins important, de s'assurer un ensemble de services de la part de son fournisseur parmi lesquels la mise en service de la solution, sa maintenance, son adaptation aux besoins spécifiques de la société, etc. Ces services ne sont possibles que tant que le fournisseur n'arrête pas ses activités ou ne décide de ne plus assurer la maintenance de la solution. La dépendance d'une société par rapport à un fournisseur peut donc devenir une source de problèmes pour la viabilité de la solution. Pour un certain nombre de secteurs de l'économie (financier, militaire ou institutions publiques) une solution opaque n'est bien souvent pas

[1] Open Simple Secure Transaction, [1] [2] [9] [13]
[2] Terme français pour désigner le Clearing
[3] EBSME : Electronic Business for SMEs [3]

légalement admissible. Le code source de la solution doit pouvoir être audité, et ce afin de pouvoir en assurer l'intégrité en terme de sécurité et de fonctionnement. Cela impose à la solution d'être en logiciel libre ou fournie avec le code source.

Parmi les solutions et protocoles existants, il n'en existe pas qui permettent de couvrir à la fois tous les besoins en terme de sécurité : indépendance vis-à-vis d'un fournisseur, disponibilité du code source et d'indépendance en terme de plate-forme. Les solutions sont soit bon marché, faciles à mettre en œuvre et incomplètes(HTTP/S[4]), soit complètes mais difficiles à déployer, complexes et onéreuses(SET[5]), soit trop exotiques et non fondées sur des standards(eCach[6]) L'analyse de ce niveau de complétude des protocoles et solutions met en exergue que la sécurisation des transactions est axée soit sur la couche transport (HTTP/S), soit sur le message lui-même (OpenPGP[7]) [10] [12]

Cet article présente le développement et l'expérimentation d'un composant en logiciel libre OpenSST qui apporte une meilleure réponse aux besoins de sécurité transactionnelle. L'expérimentation porte sur les activités de compensation appliquées aux transactions de paiements électroniques. En effet, au moment de l'achat, une demande d'autorisation de paiement est uniquement effectuée.

2. LE PROTOCOLE OPENSST

OpenSST a été développé pour répondre à trois exigences majeures. La première est l'atteinte d'un niveau de sécurité élevé en terme de confidentialité, d'intégrité, d'authentification et de non-répudiation. La seconde est la simplicité requise en ingénierie logicielle. La troisième est l'ouverture du protocole à l'acceptation et à l'utilisation de standards existants, de façon à assurer un interfaçage aisé aux d'autres solutions. Pour atteindre ces exigences, OpenSST a été bâti sur un format de message standardisé, ce qui se décompose d'une part par une structure de message, et d'autre part par le fonctionnement du protocole.

2.1 Format du message OpenSST

Le format de message OpenSST est basé sur une structure simple et adaptable. Il se présente sous la forme d'un document XML qui confère les caractéristiques d'être évolutif et portable. Trois éléments composent le message : l'*encrypion*, les *données* et la *signature*.

```
<?xml version="1.0" encoding="UTF-8"?>
<opensst version="…" xmlns="…" …>
        <encryption type="…" …/>
        <data destination="…" …>
                …
        </data>
        <signature type="…" …>
                …
        </signature>
        <tid server="…" client="…"/>
        <timestamp type="…"/>
</opensst>
```

Figure 1: format du message OpenSST (simplifiée)

L'élément *encryption* apparaît de 0 à n fois dans un même message et définit l'algorithme cryptographique, les méthodes de *padding* et le mode de fonctionnement en tant que système cryptographique hybride ou non. L'élément *données* n'est présent qu'une fois par message et contient les données transmises par la transaction. Les différents attributs de cet élément définissent entre autre : le type d'encodage de l'élément et le type de message. Le chiffrement du message étant effectif au

[4] HTTP/S opère au niveau de la couche transport et non contenu de la session. Bienqu'il supporte la non-répudiation de la session elle-même, il ne supporte pas la signature des transactions. Hors, cette fonction est primordiale pour certains sites d'eCommerce. HTTPS/S reste donc utilisé principalement dans un mécanisme d'authentification unilatérale.

[5] SET est le produit de l'association de Visa et MasterCard. Il gère exclusivement les transactions de payement. La sécurité forte garantie par SET repose sur de nombreuses techniques et algorithmes de cryptographie telles que MD5, SHA, Dual signatures, RSA et sur la séparation du contenu de l'achat et de la transaction financière. SET présente cependant les inconvénients d'un protocole lent, complexe et difficile à mettre en œuvre. Le prix de son implémentation et sa rigidité en font un outil à destination de grands comptes.

[6] eCash est système de paiement lancé par DigiCash qui repose sur un modèle de payement par « pièces de monnaie électroniques ». Il présente les inconvénients d'être lent, propriétaire et d'imposer aux couples vendeur/acheteur de posséder un compte dans la même banque eCash.

[7] OpenPGP impose un format de message binaire qui ne peut être adapté pour une application spécifique.

niveau de l'élément *données*, ce dernier peut également intégrer des éléments tels que *tid*, *timestamp* ou autre en fonction du type du message lui-même. L'élément *signature* peut apparaître de 0 à n fois et contient la signature de la partie data. Les différents attributs de cet élément définissent le type et la méthode de signature. Le strict minimum à respecter pour une transaction est décrit dans le schéma XML OpenSST basique.

2.2 Type de messages OpenSST

Différents types d'attributs de messages sont définis. Le message *createsession* initie la transaction. Il s'assure de la distribution de la clé publique du serveur au proxy OpenSST, de la création et de l'échange de clés de session, et de l'identification du serveur auprès du proxy. L'initiation de la transaction passe par les étapes suivantes : envoi d'un message par le proxy avec un minimum de données au serveur pour obtenir sa clé publique[8] ensuite, génération et transmission au serveur d'une clé de session par le proxy[9][10]. La transaction initiée, tous les messages entre le proxy et le serveur sont chiffrés avec la clé de session. Le message avec l'attribut type *KeyEnrollment* a pour fonction de fournir au serveur OpenSST la clé publique du client. Cette opération n'est effectuée que lors de la première transaction entre le client et le site proposant le paiement électronique. En partant du principe que le client a reçu de manière sécurisée du serveur un *userid* et un *pincode*, le proxy génère une paire de clés et envoie au serveur par un message type à *KeyEnrollment* contenant : *userid*, clé publique et *OTP*[11]. Le serveur répond ensuite au proxy en retournant le résultat du *HMAC*.

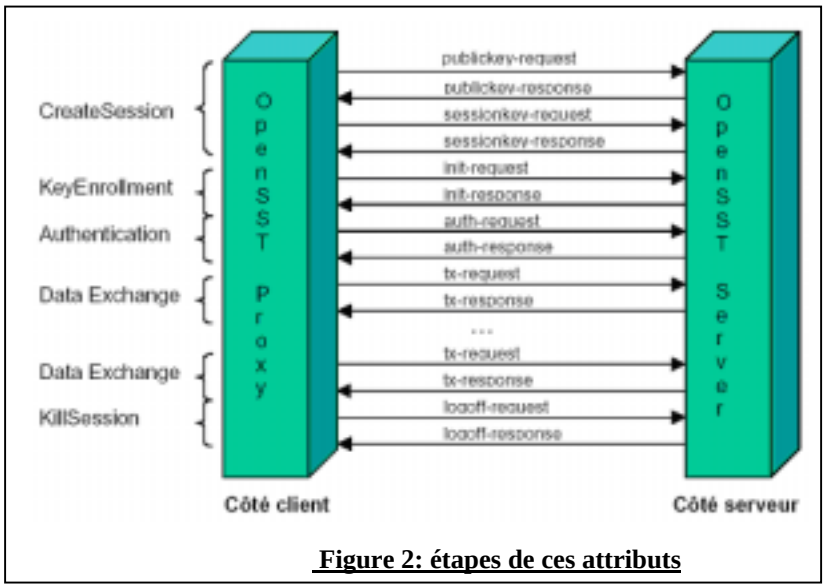

Figure 2: étapes de ces attributs

L'authentification du proxy doit être réalisée à chaque ouverture de session. Pour ce faire, le proxy transmet au serveur un message *authentification* signé qui contient le *userid*. Le serveur recherche dans sa base de données la clé publique correspondant à ce *userid* et vérifier la signature puis répond à ce dernier en lui communiquant le résultat de l'authentification.

Une fois l'utilisateur authentifié, des messages[12] peuvent être échangés entre le proxy et le serveur (messages type *dataexchange*). A chaque requête HTTP effectuée par le browser, le proxy génère un message qu'il envoie au serveur et reçoit en réponse de celui-ci un message qui contient la page Web demandée. Une requête pour une simple page Web peut générer plusieurs requêtes sous-jacentes pour obtenir images, feuilles de styles, etc.

Finalement le message *KillSession* met fin à la session. Le client envoie juste un message avec l'attribut type *KillSession* vide au serveur. La réponse de celui-ci contient un code de succès ou d'erreur et un

[8] L'attribut type de l'élément data est mis à sessionSetup1
[9] Clé de session chiffrée avec la clé publique du serveur
[10] L'attribut type de l'élément data est mis à sessionSetup2
[11] OTP = one-time password : HMAC (pincode, clé publique)
[12] Requêtes http, réponses HTML (majeure partie des cas) mais aussi des images, des CSS, Javascripts, etc.

message pour communiquer le résultat de la fermeture de session au proxy.

3. EXPERIMENTATION DU PROTOCOLE OPENSST

3.1 Application e-Business en ligne

Le déploiement d'OpenSST dans le cadre de transactions e-Business en ligne s'appuie sur trois éléments clés : un proxy OpenSST, un reverse proxy OpenSST, et un serveur OpenSST [6]

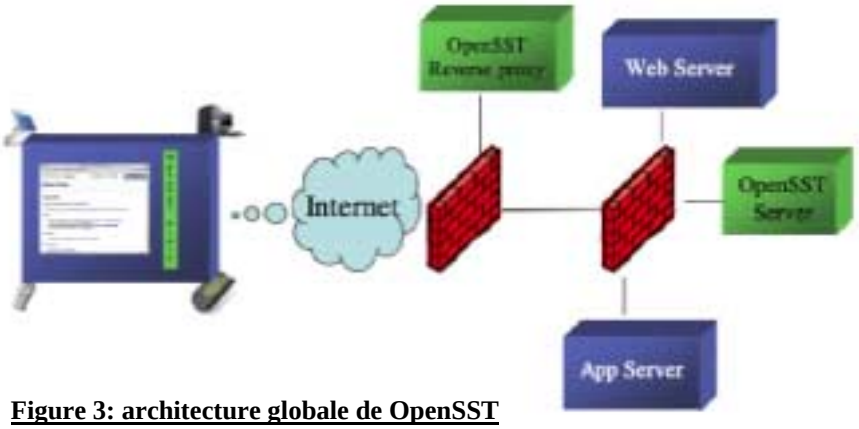

Figure 3: architecture globale de OpenSST

Les transactions transitent entre ces éléments par le protocole HTTP. Le proxy est installé sur le browser du client. Il intercepte les requêtes effectuées par ce dernier, les convertit au format et les transmet au reverse proxy. Le reverse proxy, intercepte les messages provenant du proxy, en vérifie la syntaxe mais pas la sémantique pour s'assurer que le message est bien formé. Si oui, il est transmit au serveur. Si non, le proxy reçoit un message contenant un code erreur. Le serveur reçoit le message valide, il désencapsule la requête HTTP initiale et la transmet au serveur Web. Ce serveur ne s'aperçoit pas des mécanismes de sécurité mis en place. L'opération est complètement transparente pour le client et pour le commerçant.

3.2 Domaine d'application

La compensation est un procédé permettant de centraliser et d'organiser des opérations de liquidation des dettes et créances entre 2 entités [4] [5] [7] Une société de compensation est à la fois "notaire" des opérations (conservation des traces pour ses clients en cas de contestation) et "facteur" (transport de la valeur financière quand ils changent de mains) Grâce à ces sociétés, certains porte-feuilles de titres peuvent changer de main trente fois par jour et voyager dans une dizaine de places financières. Pour le paiement électronique, la compensation peut donc être subdivisée en deux activités : Niveau 0, les marchands n'effectuent au moment de l'achat que des demandes d'autorisations de paiement. C'est à dire, que le marchand demande à un intermédiaire, en l'occurrence à une société de compensation, pour savoir si le client dispose du budget nécessaire pour

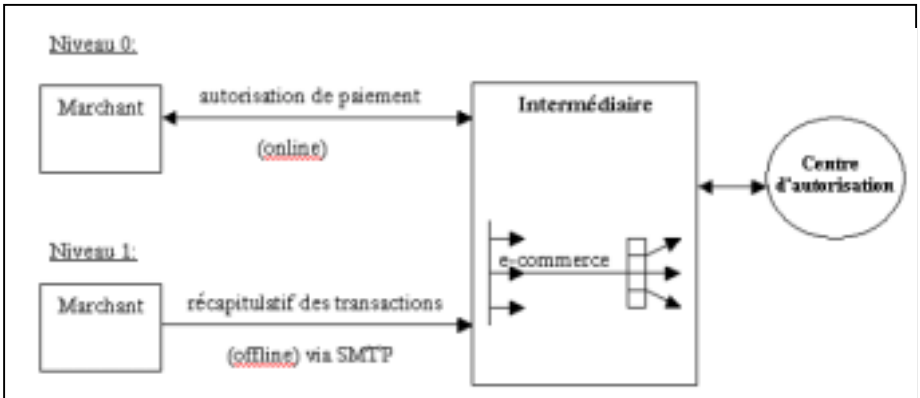

Figure 4 : vue générale d'une architecture de «Clearing »

l'achat ou non et ce durant une période d'activité déterminée. Pendant toute cette activité le marchand mémorise les divers achats via un système d'archivage des transactions en local. Ensuite, au Niveau 1, le marchand envoie périodiquement un récapitulatif de tous les achats effectués par son intermédiaire. Ce dernier s'occupera d'effectuer le transfert d'argent des comptes des divers clients vers le compte du marchant.

Cette procédure se déroule en plusieurs étapes. Pour illustrer au mieux ces étapes prenons le cas d'une personne qui effectue un achat en ligne. Les informations du client collectées par le commerçant sont envoyées à une société de compensation qui vérifie si le montant de l'achat dépasse ou non une certaine limite. Si non, la demande d'autorisation est acceptée. Si oui, l'autorisation est délivrée au non selon le type de compensation : nationale ou internationale. On parle de compensation nationale lorsqu'il s'agit d'une une carte bancaire émise dans le même pays qu'où est traitée la demande d'autorisation de paiement - liens directs avec les banques nationales. On parle de compensation internationale lorsqu'il s'agit d'une carte émise dans un autre pays que celui où la transaction est effectuée - liens par un intermédiaire international qui délègue la demande à la banque émettrice de la carte.

Les vérifications et contrôles effectués par la banque en relation avec le compte client sont selon les pays : Existence de la carte bancaire via son numéro, date d'expiration de la carte bancaire, solvabilité du compte client ou si le compte n'est pas bloqué, dépassement du montant maximum.

Après avoir effectué ces divers contrôles, la réponse et la décision de la banque sont émises à l'émetteur de la demande d'autorisation de paiement.

3.3 Description du Prototype

Le CITI a développé un prototype qui effectue une partie de compensation d'autorisations de paiements

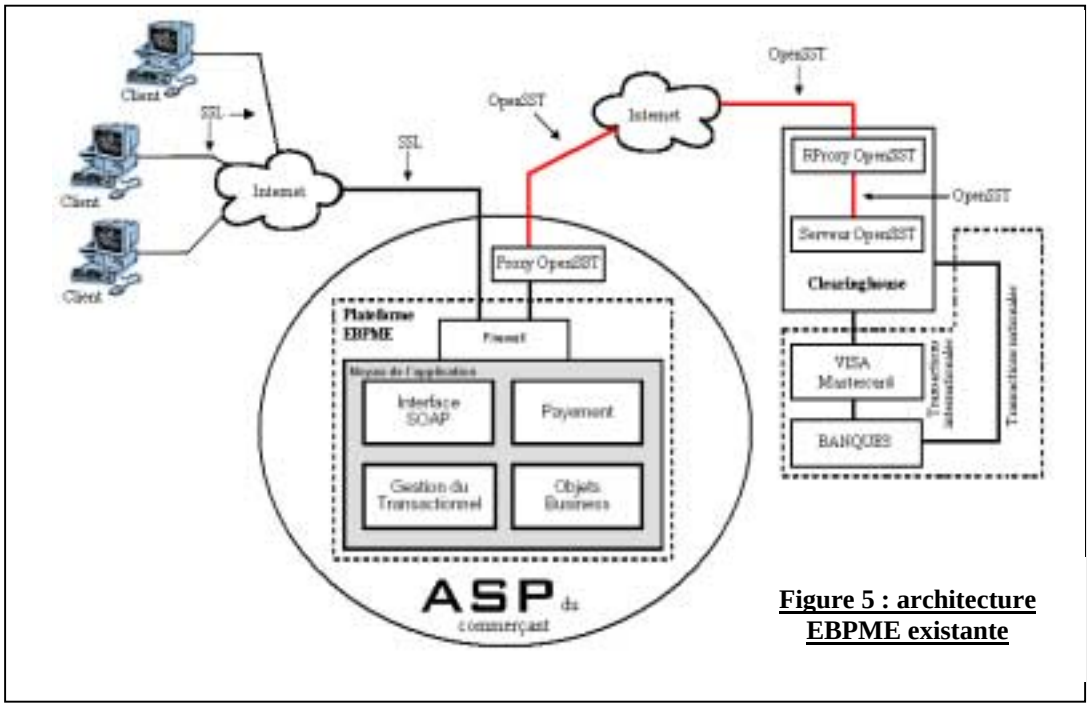

Figure 5 : architecture EBPME existante

électroniques avec cartes bancaires. La communication entre l'interface utilisateur de la plate-forme EBPME [10] et la société de compensation est sécurisée par le protocole OpenSST. La plate-forme e-

commerce EBPME a été développée dans le cadre d'un projet Fond National de la Recherche (Luxembourg). Les divers modules OpenSST tels que *Proxy*, *Reverse Proxy* et *Serveur OpenSST* qui composent le prototype sont déployés à deux niveaux :

- sur la plate-forme e-commerce EBPME hébergée par un ASP[13],
- au niveau d'une société de compensation quelconque.

La figure 5 présente les 3 acteurs : clients, commerçants et société de compensation. Les clients utilisent une liaison Internet sécurisée SSL pour communiquer avec l'application eCommerce, qui à son tour utilise une liaison Internet sécurisée via OpenSST. L'utilisation d'OpenSST impose l'utilisation d'API's qu'on retrouve greffé à la plate-forme de EBPME (Proxy OpenSST) et à la plate-forme des membres de clearing (Reverse Proxy et Serveur OpenSST). Tous les modules sont écrits en Java. De même la librairie d'encryption associé dans le cadre de la JCE est appelée Bouncy Castle jce.

Cinq nouveaux types de messages ont été développés spécifiquement pour la compensation :

- Demande d'autorisation,
- Renvoi de la demande d'autorisation,
- Réponse à la demande d'autorisation,
- Annulation de la demande d'autorisation,
- Réponse à l'annulation de la demande d'autorisation.

Ces types de messages ne représentent que ceux qui sont nécessaires à la demande d'autorisation de paiement dont la liste complète figure dans le standard ISO8583 [8]. Les divers messages sont envoyés sous forme XML dans la partie *données* du message OpenSST.

La première illustration des développements est le détail de l'élément « *données* » du message de demande d'autorisation de paiement :

```
<data destination="http://clearing_server" encoding="base64" type="e-transaction">
<msg>0100</msg>
<002>   <!-- Primary Account Number -->                    </002>
<003>   <!-- Processing Code -->                          </003>
<004>   <!-- Amount of the transaction -->                </004>
<007>   <!-- Transmission Date/Time -->                   </007>
<011>   <!-- System Trace Audit Number -->                </011>
<012>   <!-- Time, Local transaction -->                  </012>
<013>   <!-- Date, Local transaction -->                  </013>
<014>   <!-- Date, Expiration -->                         </014>
<018>   <!-- Merchant category code -->                   </018>
<019>   <!-- Acquiring Instituition country code -->      </019>
<022>   <!-- POS entry mode -->                           </022>
<025>   <!-- POS condition mode -->                       </025>
<035>   <!-- Track 2 Data -->                             </035>
<037>   <!-- Retrieval Reference Number -->               </037>
<041>   <!-- CATI -->                                     </041>
<042>   <!-- CAIC -->                                     </042>
<049>   <!-- Currency Code of the Transaction -->         </049>
</data>
```

[13] Application Service Provider

L'explication de divers tags est donnée dans ce qui suit :

msg: Permet de déterminer le type de message envoyé
002: Couramment appelé PAN, il s'agit d'un numéro variable au maximum de 19 chiffres (normalement 14 ou 16 chiffres) de longueur qui figure sur les cartes bancaires ainsi que sur la bande magnétique de ces derniers. Entres autre ce numéro permet de déterminer l'émetteur de la carte bancaire et le compte bancaire du titulaire. Ce numéro contient également un "check digit" qui permet de vérifier que la validité du compte bancaire du titulaire
003: Définie le type de transaction à effectuer.
004: Le montant réel de la transaction dans l'unité monétaire locale.
007: Date et heure de l'envoi de la transaction. Format: MMDDhhmmss
011: Le "System Trace Audit number" est un numéro allant de 000001 à 999999 qui est généré par le point de vente (POS, Point of sales) et qui reste le même pendant toute la transaction pour les messages 0100, 0101 et 0110. Le message 0400 a toujours un numéro STAN différent. Le numéro STAN est incrémenté après chaque transaction.
012: Heure locale de l'envoi de la transaction. Format: hhmmss
013: Date locale de l'envoi de la transaction. Format: MMDD
014: Date d'expiration de la carte bancaire. Cette valeur est obtenue à travers les données qui se trouvent sur la bande magnétique de la carte.
Format: YYMM
018: Contient le code qui spécifie la catégorie du marchant (l'acquéreur).
019: Code qui identifie le pays acquéreur du montant de la transaction. Pour plus amples détails sont disponibles dans le standard ISO3166.
022: Permet de savoir comment la carte bancaire a été utilisée (avec pin, sans pin,…).
025: Permet de savoir comment la transaction a été effectuée (présence du client,…).
035: Contient les informations provenant de la plage no.2 de la bande magnétique de la carte bancaire.
037: Il contient un numéro unique qui a été généré par le point de vente (POS), qui permet d'identifier une transaction. Cette valeur est nécessaire dans les messages 0100 et 0400. Ce numéro se compose de deux chiffres qui identifient la compagnie, quatre chiffres qui identifient le terminal utilisé et six chiffres qui correspondent au numéro STAN.
041: Contient un identifiant qui permet d'identifier exactement le terminal utilisé. Il est composé par le code de la compagnie (deux chiffres), l'identifiant du terminal (quatre chiffres) et l'identifiant de l'appareil utilisé (deux chiffres).
042:CAIC = Card acceptor terminal identification. Code unique qui identifie le terminal où la carte de crédit a été introduite.
049: Indique l'unité monétaire utilisée pour la transaction.

La seconde illustration des développements est le détail de l'élément « *encryption* » et de l'élément *signature* de cette même demande d'autorisation. Le tableau 1 sert à déterminer l'argument type de ces éléments. Cet argument est composé de la manière suivante pour l'encryption:

[SESSIONTYPE]-[ENCRTYPE]-[ENCRMODE]-[PADDINGMODE]-[PADDINGSESSION]-[PADDINGCRYPT]-[LENGHT]

et de la manière suivante pour la signature :

[SIGNATURETYPE]-[HASHTYPE]-[SIGENCRYPTIONTYPE]-[ENCRYPTIONMODE]-[PADDING]-[LENGHT]

ENCRTYPE	**D**	**PADDING**	**D**	**SIGNATURETYPE**	**D**	**PADDING**	**D**
Null	0	null	0	Null	0	null	0
DES	1	zero	1	Public key	1	zero	1
IDEA	2	PCKS#1 (1.5)	2	**HASHTYPE**	**D**	PCKS#1 (1.5)	2
3DES	3	PCKS#5 RFC1423	3	Null	0	PCKS#5 (RFC1423)	3
RC5	4	ISO 9796 #1	4	MD5	1	ISO 9796 #1	4
CAST	5	ISO 9796 #2	5	SHA1	2	ISO 9796 #2	5
BLOWFISH	6	ISO 9796 #3	6	**ENCRYPTIONMODE**	**D**	ISO 9796 #3	6
TWOFISH	7	PCKS#1 – OAEP	7	Null	0	PCKS#1 – OAEP	7
AES	8	IEEE OAEP	8	ECB	1	IEEE OAEP	8
ENCRMODE	**D**	ISO 10126	9	CBC	2	ISO 10126	9
Null	0	ANSI X9.23	10	CFB	3	ANSI X9.23	10
ECB	1	**SESSIONTYPE**	**D**	OFB	4	**SIGENCRYPTIONTYPE**	**D**
CBC	2	null	0	NOFB	5	null	0
CFB	3	Per-shared key	1	STREAM	6	DSS	1
OFB	4	DSS	2			RSA	2
NOFB	5	RSA	3				
STREAM	6						

Tableau 1 : paramètres de la partie encryption et de la partie signature

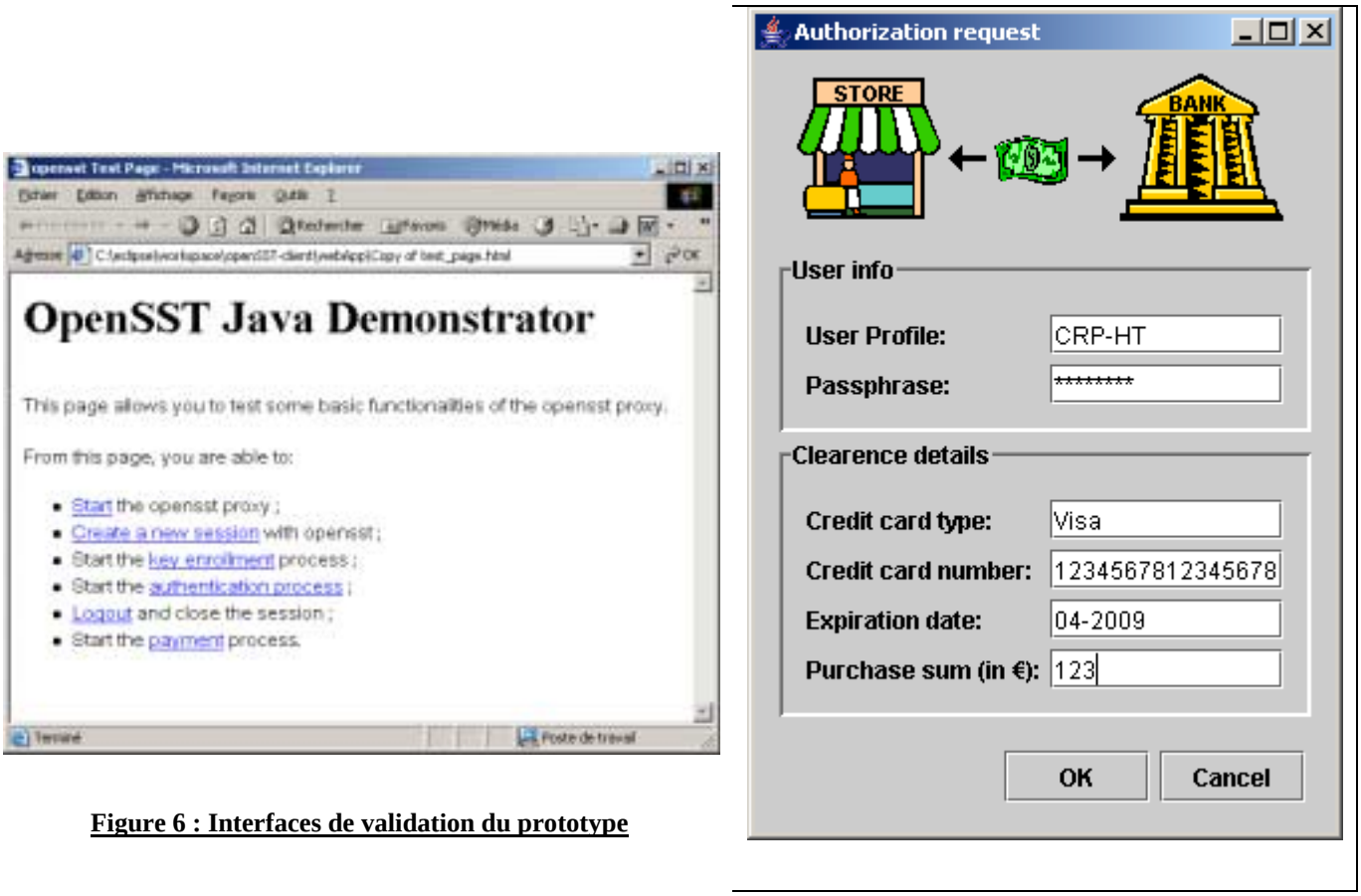

Figure 6 : Interfaces de validation du prototype

4. CONCLUSIONS

L'analyse des protocoles et solutions de sécurisation des transactions électroniques existants met en exergue qu'il n'existe pas de solution qui offre tout à la fois un haut niveau de sécurité, une assurance quant à la pérennité du système, la possibilité d'auditer le code, et un format de message malléable du protocole OpenSST. L'originalité du travail présenté dans cet article réside dans le fait que ce protocole répond à trois exigences majeures que sont l'atteinte d'un niveau de sécurité élevé en terme de confidentialité, d'intégrité, d'authentification et de non-répudiation, la simplicité par rapport à l'ingénierie logicielle et l'ouverture du protocole à l'acceptation et à l'utilisation de standards existants.

Des expérimentations menées comme celle présenté dans cet article (voir figure 6) permettent de valider et de montrer que le protocole OpenSST s'applique même à des domaines qui exigent une forte robustesse et rapidité comme le domaine du clearing. OpenSST manque actuellement d'expériences pragmatiques dans des environnements de travail réels. Sa robustesse et la qualité de son développement feront l'objet d'études complémentaires dans des projets futurs afin de l'améliorer et l'adapter aux besoins de l'informatique sécurisée.

5. BIBLIOGRAPHIE

[1] A. Dulaunoy, T. Fruru et S. Stormacq, OpenSST Message Format, Internet Drafts, December 2002.

[2] Alexandre Dulaunoy, Sébastien Stormacq - OpenSST : Open Simple Secure Transaction, Une approche de réduction de la complexité pour les transactions électroniques. SAR 2003, 30 juin – juillet 2003. Marrakech, Maroc.

[3] Djamel Khadraoui, Eric Dubois – B2B eContract Solution for Teleservices. 2003 International Conference on Intelligent Agents, WEB Technologies and Internet Commerce – IAWTIC'2003, 12-14 February 2003, page 185.

[4] Djamel Khadraoui, "OpenSST - Basic clearing mechanism for online web applications", *LinuxDays*, Luxembourg, 2003.

[5] D. O'Mahony, M. Peirce, H. Tewari, Electronic Payment Systems for E-Commerce, second edition, Artech House, 2001.

[6] Christophe Feltus, Djamel Khadraoui, "OpenSST based Clearing Mechanism for e-Business", IEEE-ICTTA'04, Damascus, Syria.

[7] http://atilf.inalf.fr/Dendien/scripts/tlfiv5/showp.exe?63;s=3375760125;p=combi.htm

[8] http://www.iso.ch

[9] http://www.opensst.org/

[10] Logiciel libre et sûreté de fonctionnement, Philippe David, Hélène Waeslynck, Lasvoisier, 2003

[11] M. Pablos Martin, T. Pinxteren, P. Robert, Sécurité du commerce électronique, http://www.tele.ucl.ac.be/ELEC2920/2000/E-Commerce/secu_et_e-commerce.html

[12] Robert L. Ziegler, *Linux Sécurité*, CampusPress, France, 2000.

[13] S. Stormacq, OpenSST Message Type : HTTP proxy, Internet Drafts, December 2002.

[14] T. Dierks, C. Allen, The TLS Protocol version 1.0, Internet Engineering Task Force, January 1999.