OpenSST based Clearing Mechanism for e-Business

Feltus, Christophe; Khadraoui, Djamel; Costa Pinto, Filipe

Published in:
Proceeding of 3rd international conference on information and communication technologies : from theory to application (ICTTA 04), Damascus, Syria

DOI:
10.1109/ICTTA.2004.1307628

Publication date:
2004

Document Version
Early version, also known as pre-print

OpenSST based Clearing Mechanism for e-Business

Christophe FELTUS*, Djamel KHADRAOUI*, Filipe COSTA PINTO*
Centre de Recherche Public Henri Tudor, 29, av. J. F.Kennedy, L - 1855 Luxembourg – Kirchberg
*Email: *{surname.name}@tudor.lu*

Abstract

OpenSST[1] is a protocol for secured electronic transactions. This article presents an experimental prototype of the OpenSST protocol within the framework of electronic transactions. It will be an e-Business architecture around which a component is graft in charge of executing authorization requests for electronic payments. This architecture is already deployed on an e-Business platform (EBSME[2]: Electronic Business for SME's). Most often a leased line ensure the clearing communication support (authorization requests). In addition to the fact of using the Internet to proceed the banking authorizations of payment, adopting OpenSST as the basic protocol for the transport of the authorization of payment messages get us the following advantages: safety, simplicity and open source. OpenSST was developed to answer three major requirements. The first requirement was to reach a high security level. This security is based in term of confidentiality, integrity, authentication and non-repudiation. The second requirement was the needed simplicity in software engineering, which proves to be a significant factor for the design of information systems. The third and the last major requirement was the ability to accept and use existing standards in order to ensure an easy interfacing with other solutions. To reach these exigencies, the OpenSST protocol was build on a standardized message format, which is composed in one part by a message structure, and on the other part by the operation of the protocol.

1. Introduction

The payment of electronic transactions is getting a continuous growth for several years. Closely related to the development of the e-commerce, the needs in transactional security, like the digital signature on the client side or the non-repudiation, seems to be critical for the use and the viability of e-business transactions over the Internet or open networks.

Several activities are achieved in the domain of electronic transactions. Indeed, the solutions and protocols are mainly of two types: proprietary or Open Source. The majority of the economical sectors[3] do not accept an opaque solution. The source code of the solution must be auditable in order to be able to ensure its integrity in term of security and operation.

This article is focused on the development of a clearing mechanism based on an open source protocol called OpenSST. It is organized as follows. The next section presents the state of the art regarding to the existing payment protocols and secure solutions of electronic transactions. The third section is dedicated to the detailed presentation of the OpenSST protocol. And the final part describing the experimentation within the framework of an e-Business clearing solution architecture and more particularly for the development of the authorization part of banking payment.

2. State of the art

In the following we will present three of the most representative solutions of the current market in the domain of the electronic transactions, namely: HTTP/S, SET and ECASH.

2.1. HTTP/S

HTTP/S (HTTP on SSL/TLS) is a protocol, which consists in securing the communication channel between two computers. It guarantees the identity of the visited website and the confidentiality of the transactions. However, operating on the transport layer, and not inside the session, doesn't make possible to check a signature when the session is closed. So, SSL doesn't support the signature of transactions, but only the non-repudiation of the session itself. If the majority of the trading websites can work without these properties, this is not the case for other applications for which this function is of primary importance. Except the transparency aspect on the application level, this protocol dissociates itself by its simplicity, performance and popularity. Nevertheless is remains used mainly in a unilateral mechanism of authentication. SET

SET is the product of the association of the two largest world organizations, which deliver credit cards: Visa and MasterCard. This protocol manages

[1] Open Simple Secure Transaction

[2] EBSME (Electronic Business for SME's) was developed within the framework of an FNR (National Fund of Research) project called ACCES-PME (Adoption de Compétences interdisciplinaires pour le Commerce Electronique Sécurisé des PME) from the CITI – The CITI, Innovation Center by Information Technologies- is a structure of the Public Research Center Henri Tudor (www.tudor.lu) dedicated to the innovation by the information systems. The CRPHT, associated with the Luxembourg University of Applied Sciences within the framework of the Campus of Technology of Luxembourg-Kirchberg, is entirely centered on the technological engineering and applications.

[3] financial, military or public institutions

exclusively the payment of transactions. The strong security (confidentiality, authentication, integrity and non-repudiation) guaranteed by SET is based on many cryptographic techniques and algorithms such as MD5, SHA, dual signatures, RSA and on the separation of purchase details and the financial transaction. Therefore the merchant can't have access to the banking data of the customer. In the same way, the bank of the merchant can't see the details of the progressing commercial transaction. However SET presents the disadvantage to be a slow and complex protocol, difficult to implement. The price of its implementation and its rigidity makes SET to be used for large accounts.

2.2. Ecash

Ecash is a payment system launched by DigiCash, pioneer in the model of electronic cash. The principle of this model rests on an order of electronic coins by a purchaser to the Ecash bank. These coins can then be spent in an anonymous way in online shops. A database updated by the Ecash bank remembers the number of coins already spent so that same coins couldn't be spent twice. On a large scale, the Ecash model could have weaknesses, more particularly time delays, caused by the large number of database accesses done during the validation of the transaction. This proprietary solution also forces the couple salesman/purchaser to have each one an account in the same Ecash bank.

Among the existing solutions and protocols, there is not solution that meets at the same time all the requirements in term of security, independence towards supplier, availability of the source code and independence in term of platform. The solutions available are either cheap, easy to implement and incomplete or complete but difficult to deploy, complex and expensive. The analysis of this level of completeness of the protocols and solutions puts forward that secure transactions are based either on the transport layer (for example the SSL/TLS or SSH protocols) or on the message itself (for example the OpenPGP protocol).

3. OpenSST protocol

OpenSST was developed to answer three major requirements. The first requirement was to reach a high level of security. This security is based in term of confidentiality, integrity, authentication and non-repudiation. The second requirement was the needed simplicity in software engineering, which proves to be a significant factor for the design of information systems. The third and the last major requirement was the ability to accept and use existing standards in order to ensure an easy interfacing with other solutions. To reach these exigencies, the OpenSST protocol was build on a standardized message format, which is composed in one part by a message structure, and on the other part by the operation of the protocol.

3.1. OpenSST message format

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

The message format of OpenSST is based on a simple and adaptable structure according to the needs of the use. It's presented in the form of a XML document, which confers the characteristics of being evolutionary and portable. The message is composed in three parts: an encryption part, a data part and a signature part.

The "encryption" element defines the used cryptographic algorithm, the "padding" methods and the operating mode as a hybrid cryptographic system or not. This element can appear 0 to N times in the same OpenSST message.

Figure 1: Message format of OpenSST

The "data" element contains the data transmitted by the transaction. The various attributes of this element define amongst other things: the encoding type of the element (for example: Base 64) and the message type (for example: specific data, a key generation message, an end of session message, etc.). The coding of the message being effective in the "data" element, this element can also integrate elements such as "tid", "timestamp" or other types according to the type of the message itself. Contrary to the "encryption" and "signature" elements, the "data" element cannot appear more than once in an OpenSST message.

The "signature" element contains the signature of the "data" part. The various attributes of this element define the type and the method of signature. The element signature can appear 0 to N times in the same OpenSST message.

3.2. OpenSST message types

Various message types are defined (figure 2), namely:

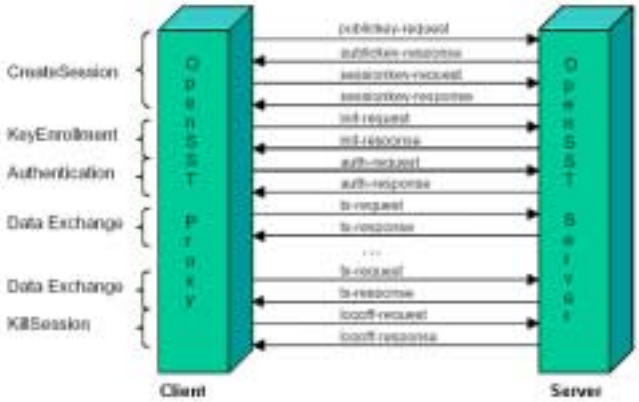
- CreateSession,
- KeyEnrollment,
- Authentication,
- DataExchange and
- KillSession.

The "CreateSession" message type aims to initialize the transaction. Among others this message sends the public key of the "OpenSST server" to the "OpenSST proxy", the creation and exchange of session keys and the authentication of the "OpenSST server" to the "OpenSST proxy". The beginning of the transaction

passes by the following stages: the sending of a message by the "proxy" to the "server" with a minimum of data to obtain its public key[4], then the generation and transmission of a session key to the "server" by the "proxy"[5][6]. After the transaction is initiated, all the messages between the "proxy" and the "server" are encrypted with the session key. The function of the "KeyEnrollment" message is to provide to the "OpenSST server" the public key of the customer. This message is send only once; the first time a transaction is done between the customer and the site proposing the electronic payment. Considering that the customer received by a secure way an "userid" and a "pincode" from "server", the "proxy" generates a pair of keys and sends to the "server" by a "KeyEnrollment" message containing the userid, the public key and a one-time password (OTP), which is a HMAC of the PINCODE and the public key[7]. The "server" answers the "proxy" by sending the result of the operation.

The authentication of the "proxy" must be carried out with each opening of an OpenSST session. With this intention, the "proxy" transmits to the server a signed "Authentication" message containing the "userid" of the customer. The server seeks in its database the public key corresponding to this "userid" and checks the signature. After this, the server communicates the result of the authentication to the "proxy". Once the user is duly authenticated, the "DataExchange" messages exchanged between the "proxy" and the "server" contains nothing else than HTTP requests and HTML responses (major part of the cases) but can also contain images, CSS, Javascripts, etc.... With each HTTP request carried out by the browser, the "proxy" sends a message to the "server" and receives in response of this a message containing the required webpage. A request for a simple webpage can generate several subjacent requests to obtain the images, the style sheets, etc.

Finally the "KillSession" message type is used, as its name indicates, to close the session. The customer sends just an empty "KillSession" message to the "server". The



response of the "server" contains a code of success or failure and a message to communicate the result of the closing of the session to the proxy.

Figure 2: Stages of the attributes schematically illustration

4. Experimentation of OpenSST

4.1. Online e-Business application

The deployment of OpenSST within the framework of e-Business transactions is based on three key elements, namely:

- an OpenSST proxy,
- a reverse OpenSST proxy,
- an OpenSST server.

The figure below shows these three elements in a global architecture.

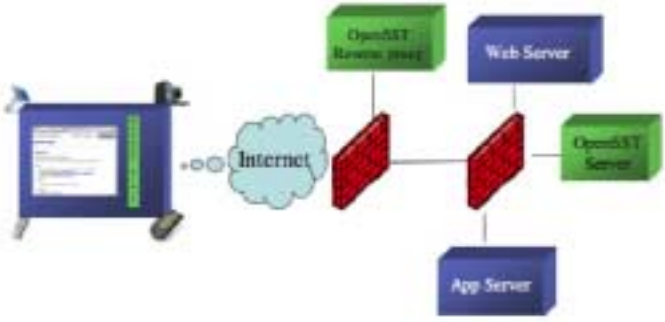


Figure 3: Global architecture of OpenSST

The transactions between these elements are forwarded by the HTTP protocol.. The "proxy" is installed on the browser of the customer. It intercepts the requests carried out by the browser, converts them to the OpenSST format and transmits them to the "reverse proxy". The "reverse proxy", generally integrated on the infrastructure of the tradesman, intercepts the OpenSST messages coming from "proxy" and checks its syntax but not its semantic to ensure that the messages are well structured. If the message format is correct, it is transmitted to the "server" for treatment. In the contrary case, the "reverse proxy" transmits to the "proxy" a HTTP message containing an error code. This can be the case for example: a message modification by an unauthorized person when it is send over the network, a malformed message done by a bugged program or a "buffer overflow" attack. The "server" receives the OpenSST message with validated syntax, decrypts the initial HTTP request and transmits it to the web server. The web server doesn't notice the used security mechanisms. The operation is completely transparent for the customer and the tradesman

[4] The "type" attribute of the "data" part is set to "sessionSetup1"
[5] Session key encrypted with the public key of the "server"
[6] The "type" attribute of the "data" part is set to "sessionSetup2"
[7] OTP = HMAC(PINcode, PuK)

4.2. Concept of "Clearing"

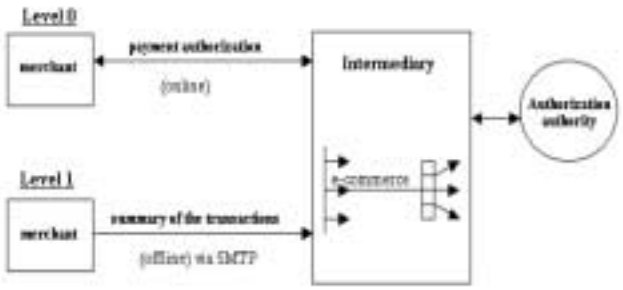
There are several definitions of the "Clearing" term:

- clearing can be most simply understood as the stage in the process after a transaction has occurred but before a portfolio has been completely adjusted and all payments (settlement) have been made.
- the verification of information between the two brokers in a securities transaction and the subsequent settlement (delivery of certificates in exchange for payment).

On the other hand, a Clearinghouse is a company, which serves at the same time as a postman, which transports the money or the values (actions or obligations) when they change the owner, and a "notary" of these operations, which carefully preserves the trace for its customers in the case of dispute. Thanks to the power of the Clearinghouses, the ownership of certain securities can change thirty times in the same day and travel in ten money markets. From a general point of view the mechanism of clearing can be subdivided in two principal levels. A schematized example of such a system is shown in the following figure.

In the case of the electronic payments (Level 0), the merchants carry out at the time of the purchase only authorization of payment requests. The merchant asks an intermediary, in this case a Clearinghouse, if the customer has or not the necessary budget for the purchase. During all this activity (over one working day for example) the merchant memorizes locally the various transactions via a backup system. Then (Level 1), the merchant sends periodically (at the end of the day for example) via mail (or other) a summary of all the purchases carried out to his intermediary. The intermediary (Clearinghouse) will carry out the money transfers from the various customers towards the account of the merchant.

Figure 4: General sight of a "Clearing" architecture



Only the "Clearing" part applied to the transactions of electronic payments will be treated in this article. Indeed, at the time of purchase, only the authorization of payment request is done.

This procedure proceeds in several stages. To illustrate these stages as well as possible, let us take the case of a person who carries out an online purchase. The collected information about the customer by the tradesman is send to a clearing service more precisely a clearing member. This member will check the amount of the purchase. If the sum doesn't exceed a certain limit, then the authorization request is directly accepted on this level.. If the sum of the purchase exceeds the imposed limit, a choice must be done according to the clearing type: national or international clearing. We are talking about national clearing when the credit card is emitted in the same country as where the transaction is carried out. Than the member has the means of treating the authorization request directly (direct bonds with the national banks). International clearing is done when the credit card is emitted in another country as the country where the transaction is carried out. In this case the clearing member that authorize the transaction request to an international intermediary (Visa, Mastercard), which delegate the request to the bank emit the credit card.

The verifications done by the customer bank during an authorization request are as follows (according to the country):

- existence of the credit card by its number,
- expiration date of the credit card,
- solvency of the client's account and checking if the client's account is not blocked
- exceed of the imposed limit

After having carried out these various controls, the response and the decision of the bank are emitted to the transmitter of the authorization request.

4.3. Prototype

The CITI has developed a prototype that will carry out part of the clearing (authorization requests) of electronic payments done by credit cards. The communication between the user interface and the EBSME platform [10] and this clearing gateway will be secured by the OpenSST protocol.

The various OpenSST modules such as the "OpenSST proxy", the "OpenSST reverse proxy" and the "OpenSST server" which make the prototype, will be deployed on two levels (see figure hereafter), namely:

- on the e-commerce platform (EBSME) hosted by an ASP[8],
- on a clearing member.

4.3.1. Global architecture

The following diagram illustrates the existing architecture of EBSME including the clearing module needed for the use of the OpenSST protocol.

On figure 5, we can find three main parts; customers, tradesmen and clearing services. The customers are using a protected Internet connection (SSL in this case)

[8] Application Service Provider

to communicate with e-commerce application, which uses an Internet connection secured via OpenSST (object of our experimentation). The use of OpenSST imposes the use of API's in relation with the EBSME platform (OpenSST proxy) and with the platform of the clearing member (Reverse OpenSST proxy and OpenSST server). All the modules are written in Java. In the same way the used encryption library "Bouncy Castle" provider is associated within the "JCE" framework.

4.3.2 Clearing messages: content of data part

We distinguish five principal message types to send to the server of the clearing bank, namely:

- authorization request,
- authorization request repeat,
- authorization request response,
- authorization reversal request,
- authorization reversal request response.

These message types represent only those, which are necessary for the authorization request of payment whose complete list appears in the ISO8583 [12] standard. The various messages are sent in XML form in the "data" part of the OpenSST message. In the figure 5, we present the contents of the "data" part for the authorization request message types for online payments.

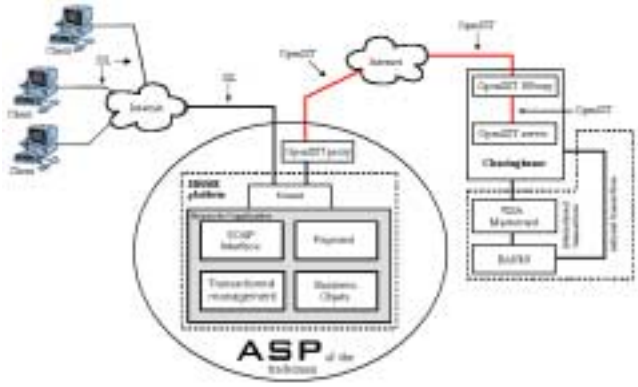


Figure 5: Diagram of the architecture of EBSME

<datadestination="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 Institution 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>

Explanation of the various tags:

msg: Allows determining the message type
002: Usually named PAN, it's a variable number with a maximum of 19 digits (usually 14 or 16 digits) retrieved from magnetic cards. This number allows determining the emitter of the credit card and the account of the cardholder. This number also contains a "check digit" which makes it possible to check the validity of the account from cardholder.
003: Defines the transaction type to carry out.
004: The real amount of the transaction in the local currency.
007: Date/Time of the transmitted transaction. Format: MMDDhhmmss
011: The "System Trace Audit number" is a number between 000001 and 999999 that is generated by the point of sales(POS) and which remains the same during all the transaction (messages 0100, 0101 et 0110). The **0400** message has always a different STAN number, which is incremented after each transaction.
012: Local transaction time. Format: hhmmss
013: Local transaction date. Format: MMDD
014: Expiration date of the credit card. This value is retrieved from magnetic card. Format: YYMM
018: Merchant category code.
019: Code that identifies the country of the purchaser. For further details are available in the ISO3166 standard.
022: Allows knowing how the credit card was used (with or without pin-code).
025: Allows knowing how the transaction was carried out (presence of the customer…).
035: Contains the information from the track no.2 of the magnetic tape of the credit card.
037: It contains a unique number generated by the point of sales (POS), which makes it possible to identify a transaction. This value used in the 0100 and 0400 messages. This number is composed of two digits, which identify the company, four digits, which identify the used terminal, and six digits that correspond to STAN number.
041: Contains a value, which identifies the used terminal. A "company code" (two digits), the terminal identifier (four digits) and the identifier of the used device (two digits) compose this value
042: CAIC = Card acceptor terminal identification. It's a unique code that identifies the terminal where the credit card was introduced.
049: Indicate the monetary currency used for the transaction.
The various encryption algorithms and supported signatures within the framework of this experimentation and the OpenSST prototype are gathered in the sections below

4.3.2 Clearing messages: content of encryption part

This table is used to determine the argument "type" of the element "encryption" of the OpenSST message. The argument "type" is made up in the following way:

[SESSIONTYPE]-[ENCRYPTIONTYPE]-[ENCRYPTIONMODE]-[PADDINGMODE]-[PADDINGSESSION]-[PADDINGCRYPTION]-[LENGHT]

ENCRYPTIONTYPE	ID		PADDING	ID
Null	0		Null	0
DES	1		Zero	1
IDEA	2		PCKS#1 (1.5)	2
3DES	3		PCKS#5 (RFC1423)	3
RC5	4		ISO 9796 #1	4
CAST	5		ISO 9796 #2	5
BLOWFISH	6		ISO 9796 #3	6
TWOFISH	7		PCKS#1 – OAEP	7
AES	8		IEEE OAEP	8
			ISO 10126	9
ENCRYPTIONMODE			ANSI X9.23	10
Null	0			
ECB	1		SESSIONTYPE	
CBC	2		Null	0
CFB	3		Per-shared key	1
OFB	4		DSS	2
NOFB	5		RSA	3
STREAM	6			

4.3.2 Clearing messages: content of signature part

This table is used to determine the argument "type" of the element "signature" of the OpenSST message. The argument "type" is made up in the following way:

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

SIGNATURETYPE	ID		PADDING	ID
Null	0		Null	0
Public key	1		Zero	1
			PCKS#1 (1.5)	2
HASHTYPE			PCKS#5 (RFC1423)	3
Null	0		ISO 9796 #1	4
MD5	1		ISO 9796 #2	5
SHA1	2		ISO 9796 #3	6
			PCKS#1 – OAEP	7
ENCRYPTIONMODE	ID		IEEE OAEP	8
Null	0		ISO 10126	9
ECB	1		ANSI X9.23	10
CBC	2			
CFB	3		SIGENCRYPTION TYPE	
OFB	4			
NOFB	5		Null	0
STREAM	6		DSS	1
			RSA	2

5. Conclusions

The analysis of the existent protocols and solutions for secure electronic transactions puts forward that it doesn't exist a solution which offers all at the same time, a high level of security, an insurance for perennial systems, the possibility to audit the code and a malleable message format.

As a consequence of what was mentioned above, a partnership of companies[9] developed the OpenSST protocol. This protocol answers three major requirements that are:

- a high level of security in term of confidentiality, integrity, authentication and non-repudiation,
- the simplicity compared to software engineering,
- and the ability to accept and use existing standards.

Experiments carried out and presented in this article show that the OpenSST protocol can be applied even to domains, which require a strong robustness and speed like the clearing domain. OpenSST robustness and the quality of its development will make complementary researches and implementations in various future projects in order to adapt it to the needs of secured information technology.

6. References

[1] T. Dierks, C. Allen, “The TLS Protocol version 1.0”, *Internet Engineering Task Force*, January 1999.

[2] A. Dulaunoy, T. Fruru et S. Stormacq, “OpenSST Message Format”, *Internet Drafts*, December 2002.

[3] S. Stormacq, “OpenSST Message Type: HTTP proxy”, *Internet Drafts*, December 2002.

[4] Proposed : A. Dulaunoy, S. Stormacq, “OpenSST : Open Simple Secure Transaction : Une approche de réduction de la complexité pour les transactions électroniques”, January 2003.

[5] Ph. Oechslin, Quelques notions de cryptographie, http://lasecwww.eppfl.ch/securitereseaux/files/s1_07.pdf

[6] M. Pablos Martin, T. Pinxteren, P. Robert, Sécurité du commerce électronique,

[9] Aubay Luxembourg SA, Conostix SA and the CRP Henri Tudor

http://www.tele.ucl.ac.be/ELEC2920/2000/E-Commerce/secu_et_e-commerce.html

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

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

[9] Alexandre Dulaunoy, Sébastien Stormacq - OpenSST : "Open Simple Secure Transaction, Une approche de réduction de la complexité pour les transactions électroniques". *SAR 2003*, 30 July – June 2003. Marrakech, Maroc.

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

[11] CEntre de TRansferts Electroniques: http://www.cetrel.lu

[12] International Organization for Standardization: http://www.iso.ch

[13] Alexandre Dulaunoy, Sebastien Stormacq, 2002. "OpenSST: Open Simply Secure Transaction". URL: http://www.foo.be/current/opensst/

[14] Alexandre Dulaunoy, April 2002. "A XML Schema for the OpenSST message format". Internal Conostix document.

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

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