Authentication and enterprise secured data storage

Nathanael Cottin
SeT Laboratory
University of Technology
of Belfort – Montbeliard
90010 Belfort
France

Bernard Mignot
SeT Laboratory
University of Technology
of Belfort – Montbeliard
90010 Belfort
France

Maxime Wack
SeT Laboratory
University of Technology
of Belfort – Montbeliard
90010 Belfort
France

***Abstract* – Data certification and digital signature are a new area of interest and many standards have emerged. Indeed, these technologies offer identification, authentication and non-repudiation capabilities during Internet transactions (emails and e-commerce). However, it appears that both certification and digital signature do not completely answer enterprise data authentication and secured data storage needs.**

We submit a proposition of an authorities-based architecture to answer these issues. This architecture relies on most of the available standards.

I. INTRODUCTION

Electronic certification and enterprise secured data storage are a new area of interest. The main goal to achieve is to delegate documents management to storage trusted third parties (i.e. TTP) [26] in case that these documents are protected by cryptography and digital signature. This procedure implies the use of technical solution which first give legality to electronic documents (with the supply of a certification procedure) and second allow enterprise private data transfer to in accordance with individuals privacy.

Standards emerge to meet the demands for data protection as well as individuals identification and programs authentication in most areas of data communications.

Electronic certificates combined with cryptography partly answer tiers-based secured data storage. However, existing techniques suffer from their lack of large scaled application.

In this article we discuss a proposition of an authorities-based architecture. This architecture is designed to supply enterprise needs in terms of data certification and storage. It relies on most of the existing standards (X.509 [18], certificate requests [27], OCSP [28], TSP [1] and related standards).

II. CERTIFICATION PROCESS

Message transfer in general and through the Internet in particular suffers from its historical non-secured architecture. Communication may be secured by encryption protocols such as SSL [14] and PPP [21], it is often more valuable to undoubtly *identify* the sender and *authenticate* the received message than authenticate hardware or applications (web browsers for example). Thus, the receiver of a message does not have any concrete proof of the sender's identity. It may happen that a man in the middle (MITM) masquerades as the real sender of the message [21]. Neither can the receiver prove that the message he received was the message the sender intended to communicate.

These two common security issues may be overcome by the use of digital signature and electronic certificates.

A. Digital signature and message signing basics

Digital signature is the current way of authenticating electronic data. It is the achievement of many research on asymmetric key cryptography and hashcoding.

A1. Asymmetric key cryptography concepts

When a sender entity (a person, a server or a program) needs to securely send a message to a receiver entity, it encrypts the message using the receiver's public key. This key is published so that any sender can make use of the receiver's public key to encrypt data. The encrypted message is then unintelligible and cannot be decrypted without the corresponding private key. This private key must be securely stored by the receiver that does not publish it. Only the receiver would then be able to decrypt the encrypted message. Asymmetric key cryptography achieves *privacy* and *confidentiality*.

The most widely used asymmetric key cryptography algorithms are RSA [35] and triple-DES [32].

A2. Hashcoding overview

Hashcoding [26] aims at creating a fixed-length message digest from any arbitrary-length data stream. This digest is size-independent of the size of the source stream. Let's consider *h()*, a one-way *hash* function used to compute a digest on a given stream *s*. The most important property of this function prevents source stream reconstruction only if the computed digest is known. Although reconstructing the original stream *s* from a given digest *d* may theorically be possible, it appears to be computationally unfeasible:

$$\left(h(s)=d\right)\Rightarrow\left(p\left(h^{-1}(d)=s\right)\rightarrow 0\right).$$

Moreover, the probability *p* that two different streams s_1 and s_2 obtain identical digests with a given hashcoding algorithm *ha* reaches zero. The hash function is then said *collision resistant*:

$$\left(s_1\neq s_2\right)\Rightarrow\left(p\left(h(s_1,ha)=h(s_2,ha)\right)\rightarrow 0\right).$$

Many digest algorithms such as MD2 [22], MD4 [40] and RIPEMD [7] [36] have been developped. The most popular algorithms SHA-1 [30] and MD5 [41] are specifically designed to compute digital signatures.

A3. Digital signature

Digital signatures defined by [33] reproduce waxed seals used in the Antiquity to seal letters up.

The seal may be compared to a *secret signature key* that should only be in possession of the *signer*, i.e. the entity that signed the message. Although a seal remains identical independantly of the letter information, the digital signature is message-dependent. This means that applying a signature key (the signer's private key) on two different streams will result on two different digital signatures. On the contrary, the same stream will always generate the same signature in case a given signature algorithm is used. However, the signer's unique corresponding *verification key* (its public key) may be used to make sure the signature has been computed using its signature key.

Digital signatures generation is nothing but the application of asymmetric key cryptography over streams hashcodes. Unlike data encryption, digital signature's purpose does not consist of data confidentiality but rather in providing [21]:

- *Data integrity*: digital signatures allow to detect source streams alteration, i.e. unauthorized data modification
- *Authentication*: as the signature key is (theorically) owned by the signer only, it is impossible for anyone else to generate the sender's signature on a given data stream. The stream is authenticated by comparing the signature with the signer's corresponding verification key
- *Non-repudiation*: this authentication-based service is a proof of transaction effectiveness. The signature entity cannot deny being the author of the signature because nobody else could possibly have created such a signature on a given data stream.

Digital signature is generally computed on hashcodes rather than directly using data source streams. The main reason is that digital signature is more time and processor consuming than the hashcoding process. It is then profitable to apply digital signature generation algorithms (DSA [33] and ECDSA [20] [2] for example) on hashcodes.

Although digital signature makes it possible to authenticate received data, it does not *identify* the entity that signed the data (the *signer*) from the receiver point of view. Thus, any irrefutable link exists between the signer and its signature key. Such identification is provided by *electronic certificates*.

B. Electronic (qualified) certificates

An *electronic qualified certificate* (i.e. certificate) is an electronic proof of identity (fig.1). It is designed to allow senders *identification* by signed messages receivers. Certificates trust depends on their issuers trust. Only Certificate Authorities (CAs) are considered as TTPs in PKIs which rely on the X.509 standard [25]. Other entities are not accredited by governments to deliver electronic certificates. Once an adequation between an entity and a signature key has been demonstrated, a qualified certificate is issued.

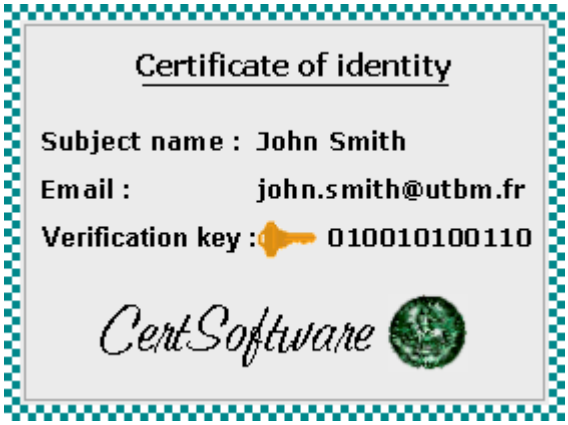

Fig.1 Certificate common description

Each certificate is identified by its unique serial number given by the issuing Certificate Authority (CA). Indeed, electronic certificates' primary rule is to associate a signature verification key and a signer. Depending on the signature key rules, they may be used for:

- *Secure emails*: certificates may be integrated within secure email standards such as PGP [16] [5] [10], PEM [24] [23] [3] and S/MIME [37] [38]
- *Code-signing*: Java Archives [42] [13] and Microsoft Authenticode [17] make the most of code certification needs
- *Identify parties*: during Internet transactions, end-entities may be identified by decoding their digital signatures with their verification key. This key is enclosed in their certificates.

Certificates are valid until they are revoked or until they expire (fig.2). In both cases a new certificate may be re-issued by the CA. A certificate revocation occurs when its owner is aware that the certificate is corrupted or that a non-authorized entity may have used it. It may also be possible for the government or the CA to revoke a certificate in case its owner makes fraudulent use of it.

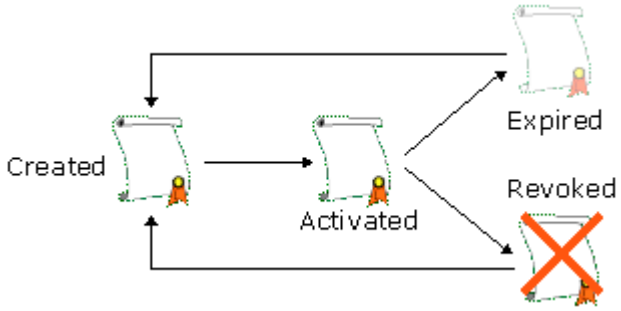

Fig.2 Certificate lifecycle

Certificates and digital signatures main goals are entities (individuals, servers and programs) *identification* as well as data *authentication*. This leads to design a trusted authority responsible for secured data storage considering that such a TTP may *securely* and *legally* store enterprise data.

III. ARCHITECTURE OVERVIEW

We designed an authorities-based architecture (fig.3) where each authority plays a key role to authenticate data and certify documents and identities (individuals, enterprises, servers or programs). The starting point of this architecture is the CA which delivers electronic certificates. It is crucial that certificates delivery assures that any certificate is given to the right person and that the enclosed information is valid and verified. The CA may be the weakest authority in case secured certificate delivery protocols (SCDPs) are not supplied. In such a case, digital signature would lose its legal scope considering that signatures rely on certificates' trustworthiness.

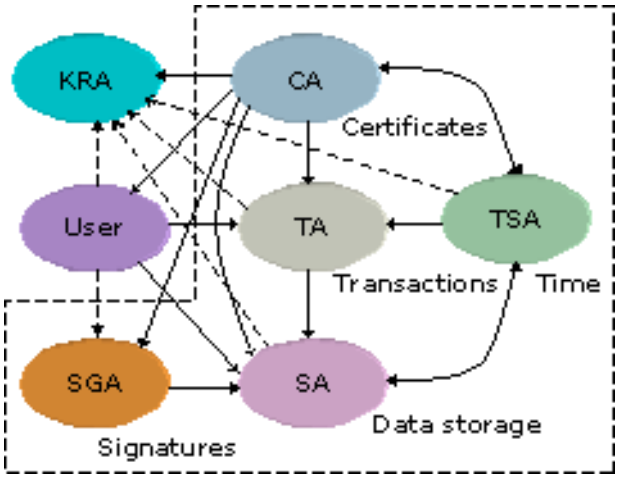

Fig.3 Architecture overview

Assuming that certificates are delivered without any possible corruption or fraud, the architecture is articulated around five other authorities (all of them are TTPs): Timestamping Authorities (TSAs), Signature Authorities (SGAs), Key Recovery Authorities (KRAs), Storage Authorities (SAs) and Transactions Authorities (TAs). The last two authorites will be discussed within dedicated sections.

A. Timestamping Authority

A Timestamping Authority (TSA) attributes a legal time value to a given message digest. Thus, a signature has no legal value if not timestamped because there would be no way to control that the signature was created while the signer's certificate was valid (not revoked or expired). It is also used to authenticate a document and repudiate a fraudulent copy of it – in case the original document was signed before the copy.

As an extension of the digital signature structure presented by [18] and expressed with the ASN.1 [8] notation, a legal signature includes a timestamp based on the Time Stamp Protocol (TSP).

It may be composed of the following information:

```
Module-LegalSignature
DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

LegalSignature ::= SEQUENCE {
  signature           Signature,
  -- of a document
  timestamp           Time,
  -- given by a TSA
  tsaSign             TSASignature
}

Signature ::= SEQUENCE {
  algorithmId         Identifier,
  policy         [0]  IMPLICIT   Policy   OPTIONAL,
  sign                BIT STRING,
  issuer              Issuer
}

Time ::= CHOICE {
  utcTime             UTCTime,
  generalTime         GeneralizedTime
  -- SHOULD be preferred to UTCtime (from TSP)
}

TSASignature ::= SEQUENCE {
  sign                Signature,
  -- signature of the TSA
  accreditations      Accreditations  OPTIONAL
  -- requested accreditations
}

Policy ::= Identifiers

Identifiers ::= SEQUENCE OF Identifier

Identifier ::= SEQUENCE {
  id                  OBJECT IDENTIFIER,
  parameters          ANY  DEFINED BY id     OPTIONAL
}

Issuer ::= ReqCert
-- ReqCert defined by OCSPv2
-- to make it OCSPv2 compliant

ReqCert ::= CHOICE {
  certID              CertID,
  issuerSerial   [0]  IssuerAndSerialNumber,
  pkCert         [1]  Certificate,
  -- as described in RFC2459
  name           [2]  GeneralName,
  -- as described in RFC2459
  certHash       [3]  OCTET STRING
}

CertID ::= SEQUENCE {
  hashAlgorithm       AlgorithmIdentifier,
  -- as described in RFC2459
  issuerNameHash      OCTET STRING,
  issuerKeyHash       OCTET STRING,
  serialNumber        CertificateSerialNumber
  -- as described in RFC2459
}
```

```
IssuerAndSerialNumber ::= SEQUENCE {
  name                RDNSequence,
  serial              CertificateSerialNumber
  -- as described in RFC2459
}

Accreditations ::= SEQUENCE OF   Signature
-- signatures of Accreditors
-- Accreditors check that the proposed
-- timestamp is valid (under a policy-specified
-- time delay).

CertificateSerialNumber ::= INTEGER
-- certificate serial number (from RFC2459)

END
```

In the previous structure, *Signature* data represent the commonly-speaking signature [18] [29]. It is composed of the signature itself and the signer's certificate identifier (composed of the certificate's serial number and issuer's name). This identifier is used to get the certificate's information for the issuing CA and make sure the signature information match the signed data.

Moreover, *LegalSignature* represents a computed signature on any data (a message) given by another party (a SA or a TA for example). This signature is completed with a timestamp and may be countersigned by Timestamping Accreditation Authorities – or Accreditors – (fig.4). This makes sure the proposed timestamp is valid on any given timestamping policy [12].

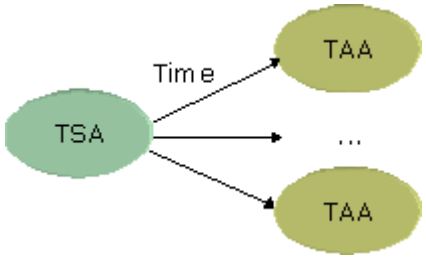

Fig.4 Use of timestamp accreditors

The timestamp request is given by the TSA via *TimeStampReq* information proposed by [1].

The *TSASignature* data structure may be integrated within pkcs#7 [35] signed-data component and TSP. The interest of using such a definition of a legal signature is that it both integrates timestamp tokens [1] and OCSPv2 [29] online verification abilities.

Accreditors are appealed by TSAs via *accreditation requests*. Such data structures and protocols are beyond the scope of this article and will not be presented.

The main issues concerning such authorities are firstly maintain a unique global clock when multiple TAs and Accreditors are present and secondly provide a secured timestamping environment. This is part of [19] and [12] recommandations.

B. Signature Authority

The Signature Authority (i.e. SGA) tackles the issue of allowing multiple signers of a given document. Indeed, SGAs act as electronic notaries which supervise the signing process by collecting the different signatures. Considering that any signer would trust the other signers, SGA will be given the ability to store (and cipher) the document. Different signing policies are in progress to complete [11] definitions.

An enterprise document may be a spool and be signed by more than one person (contracts, bills, for example). As it may have multiple owners, document signing, consultation and removal protocols have to be defined.

The main goal to achieve is to allow multiple signers to sign up data over the Internet. We are currently designing a protocol to handle this issue. This protocol is based on pkcs#7 signed-and-enveloped-data definitions. The basic idea is that Signature Authorities supervise and manage the multiple signature processes.

C. Key Recovery Authority

The Key Recovery Authority (i.e. KRA), also known as Key Escrow or Trust Center, is requested by government institutions so that they can access to encrypted data. With cryptography legalization, asymmetric keys may be 2048 bits length or more and makes it practically impossible to decipher data within an acceptable time.

On the one hand, governments want to be able to decrypt all data to make sure that secret documents do not leave their country and stay secret, and non-authorized documents do not circulate within their country. On the other hand, individuals and enterprises fear for their privacy.

The latter have to give a copy of their encryption/decryption keys to Key Recovery Authorities which make them available to governments only if necessary.

IV. SECURED DATA STORAGE AND CONSERVATION

According to the law, electronic documents do not have any legal scope unless they have been digitally signed (fig.5) and the signer may be identified.

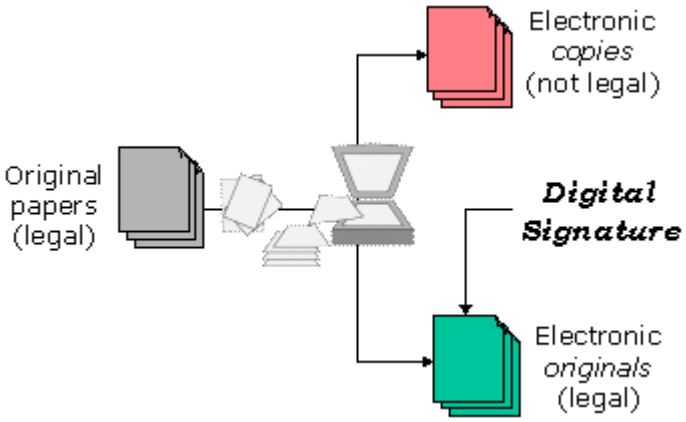

Fig.5 Digital signature utilization

Two technical issues raised here are firstly give electronic copies of a document a similar value to the original document and secondly securely store and conserve these *legalized* electronic documents. Digital signature and electronic certificates combined together partly answer the first issue by respectively providing data authentication and signers identification. However, signed documents storage within a legal context has not been considered yet.

A. Storage Authority presentation

A Storage Authority (i.e. SA) is a TTP which responds to enterprise document storage-related needs. These needs may be classified as follows:

- *Data integrity*: this service is provided by a secured storage combined with a digital signature which indicates whether a given document has been modified or not. The original document (that is the document stored at first) must be kept by the SA to make sure authorized governmental institutions have access to the first version of each stored document
- *Confidentiality*: cryptography may be used by SAs to make sure data is unreadable unless they explicitly decrypt it. This *software* protection may be completed by a *hardware* protection such as designing 3-tiers architectures
- *Access privileges*: is achieved by controlling access to stored documents. Authorized entities are governments representatives and signers only
- *Documents availability*: compared to traditional data storage, access delay to documents is less important than documents availability and authentication. It is the SA's responsability to make sure that any stored document is available to an authorized entity and that all possible actions on the document have been traced
- *Documents perennity*: SAs provide documents perpetuation considering that documents may be lifelong stored and viewed. This particular issue has not been solved yet
- *Traceability*: similarly to CAs, these authorities must establish traces for all transactions (this includes documents storage, consultation, deletion, modification – if applicable –). Traceability lays on fraud detection and diagnostic.

B. Storage Authority architecture

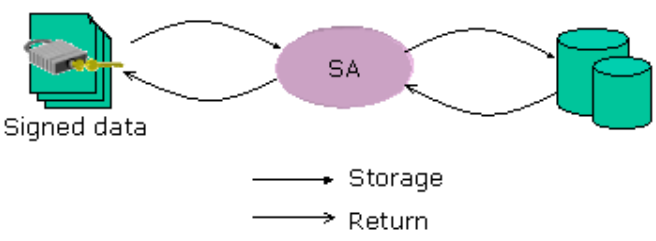

Fig.6 SA architecture overview

The SA is basically composed of one or more secured databases (fig.6) used to store signed documents and other related information which give the storage a legal aspect.

We have defined a general data structure for documents to be encapsulated within pkcs#7 base class ("just data", with no cryptographic enhancements) and stored by SAs:

```
Module-Document
DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

Document ::= SEQUENCE {
 docInfo          DocumentInfo,
 -- common document information,
 -- such as file name, title, etc.
 components     SEQUENCE OF DocComponent,
 -- one or more files may constitute
 -- the "document"
 extensions [0]  IMPLICIT  Extensions  OPTIONAL
 -- as defined in RFC2459
}

DocumentInfo ::= SEQUENCE OFIdentifiers
-- defined in Module-LegalSignature

DocComponent ::= SEQUENCE {
 componentInfo      DocumentInfo,
 -- information concerning the component
 -- such as file name, etc. to make it easy
 -- to extract
 componentData      BIT STRING
 -- and not OCTET STRING : a component may
 -- be a signature
}

END
```

V. E-BUSINESS INTEGRATION

Electronic commerce (business conducted over the Internet) is dramatically increasing around the world. E-business websites benefit from our mentalities evolution even if europeans seem to be reluctant to accept payment over the Internet.

This situation is due to the non-legal recognition of electronic commerce and the lack of redress definition for both the customer and the merchant in case of fraud or contest [15]. Particularly, *transaction repudiation* is still permitted by the law.

Thus, most of the technical issues lay during data transmission (specifically the payment) and *traceability*. In case it may happen the transmitted data is lost or corrupted, the customer will probably lose the merchant's trust.

Another minor issue is to provide a system which simplifies the customer navigation and memorizes the customer's actions every time he enters the merchant's website for purchase.

A. Transaction Authority presentation

Our framework gives an answer to this unavoidable scheme where the customer and the merchant do not trust each other. One of the reasons is that any legal Internet transaction traceability is defined. The basic idea is to integrate a *Transaction Authority* (i.e. TA) on top of e-commerce protocols such as ECML [34] and JCM [9] [4]. This authority must be recognized as a TTP by both the merchant and the customer. Each transaction is then supervised by the TA, *traced* and *stored* locally or within a Storage Authority (the transaction trace sent by the TA is then considered by the SA as a common message):

The idea is to define a merchant's community of trust (COT): each participating merchant must be labellized. This label appears on its website and ensures customers that any e-commerce operation is traced and supervised using the TA technology.

B. Transaction Authority architecture

Though Transaction Authorities deal with legal traceability within e-commerce transactions, they do not supervise and trace any kind of business. Only registered e-commerce sites within a given TA benefit from the services provided by this authority. A registered site (i.e. member) is a website hosted by the TA (fig.6).

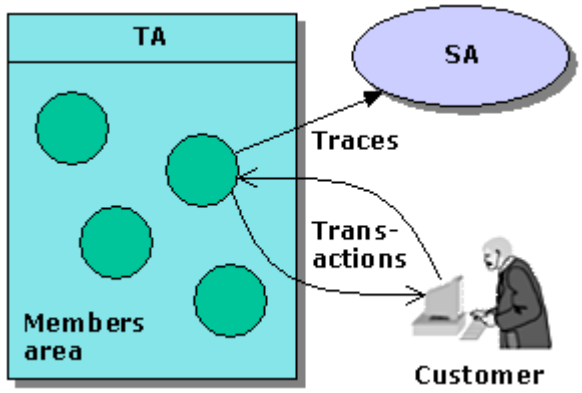

Fig.6 TA architecture overview

Becoming a member implies acceptation of the TA's hosting policy. This policy integrates *website supervision* (which may include online tests) as well as *transactions traces*.

TAs also give the opportunity to use *electronic money* (i.e. e-money) to pay without providing any credit card number to the merchant. The payment is indirectly performed by the customer when using the following purchase protocols.

C. Purchase protocols

To securely effect payment over the Internet, Transaction Authorities require that the customer buys e-money to credit his e-account. His e-account is either managed by his bank or by the TA. Hosted vendors will then accept this electronic money for payments. Two protocols based on Electronic Data Interchange (EDI) [31] [43] are combined together to allow secured e-accounts provision and electonic payments.

The first protocol (fig.7) describes the procedure to obtain e-money credits, considering that the TA manages the customer's e-account and that the customer is already registered within the TA (a customer's e-account has been created).

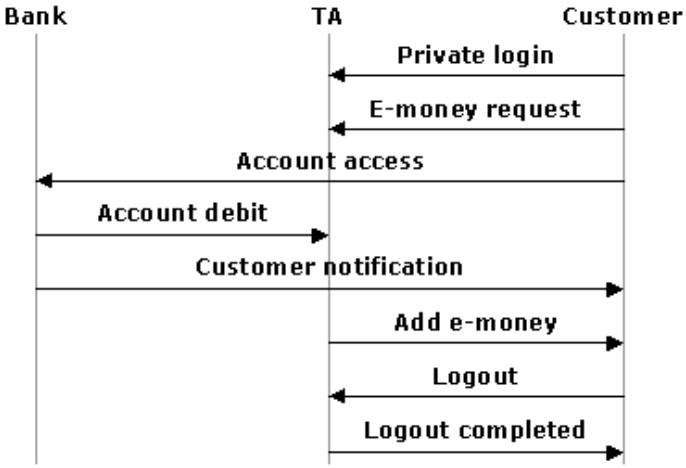

Fig.7 E-money account credit

The customer enters the TA's private area using a login name and a password (or a certificate). The TA then offers to credit the customer's e-account. The customer sends a credit request by giving its credit card number (the customer can also mention an automatic bank account debit) and the amount of e-money he wants to credit. The TA initializes a communication with the customer's bank so that the customer agrees with his bank to debit his bank account. The customer is notified by his bank that his account has been debited and then by the TA that the equivalent amount for e-money has been credited on his e-account. The customer can then make use of his e-money.

The second protocol (fig.8) describes the procedure to use e-money credits on the customer's e-account to purchase on a registered e-commerce website (member). This second protocol is fairly similar to the first protocol.

The customer logs into the member's private area. When he decides to send a purchase order to the merchant, the latter forwards the request to the TA by giving the customer's identifier. The TA checks that the corresponding e-account has enough e-money provision to perform the transaction and sends the client a debit authorization. The client then confirms the debit operation. Once the member's account is credited and the customer's notifyed of his e-account debit, the customer can then logout.

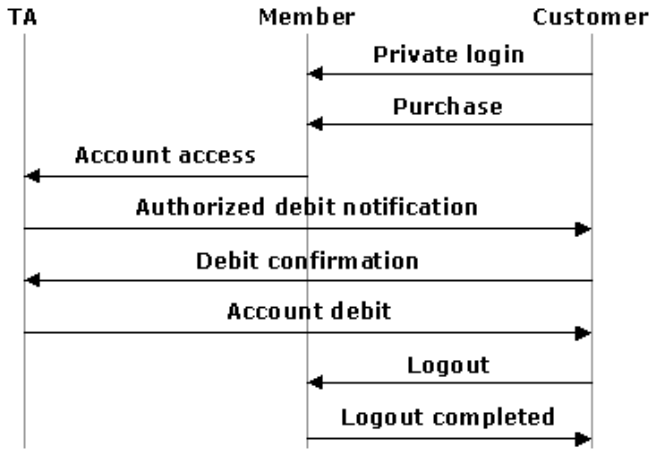

Fig.8 Purchase using e-money account

It is important to be aware that only TAs collect information about customer's identity. Access to the merchant's website is made possible through an identifier which may not be linked with the customer's identity (from the merchant's perspective).

VI. CONCLUSION AND FUTURE WORK

In this article we have presented an authorities-based architecture that we partly implemented. This architecture aims to respond to enterprise needs in terms of message sender identification and received data authentication. Its modularity resides on the distinction between multiple trusted third parties defined by the services they can afford. It is specifically designed to easily integrate new protocols and tackle scalability issues. We partly implemented a prototype based on the proposed architecture. This prototype by now includes a *Certificate Authority* – which delivers X.509v3 [18] certificates – as well as a *Storage Authority*, a *Timestamping Authority* and a *Timestamping Accreditation Authority* (Accreditor) which all conform to the proposed data structures and protocols.

We are currently working on modelling and simulating certification protocols (SCDPs), multi-signature protocols (MSPs) as well as defining Quality of Service (i.e. QoS) parameters based on our previous researches [6]. At the end of our research, it appears that new concepts such as certificates delegation ability and intrinsic data protection are necessary to anticipate enterprise future needs.

Most of the security issues over the Internet rely on the data exchange protocols supplied. Next publications will discuss our SCDPs and multi-signature protocols simulation and validation results as well as new protocols presentation.

REFERENCES

[1] C. Adams, P. Cain, D. Pinkas, R. Zuccherato, “RFC 3161: Internet X.509 Public Key Infrastructure: Time Stamp Protocol (TSP)”, August 2001

[2] American National Standards Institute, “Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm”, January 1999

[3] D. Balenson, “RFC 1423: Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers”, TIS, IAB IRTF PSRG, IETF PEM WG, February 1993

[4] A. Brown, “Java Commerce Messages: A messaging format for the Java Electronic Commerce Framework”, Sun Microsystems, 1999

[5] J. Callas, L. Donnerhacke, H. Finney, R. Thayer, “RFC 2440: OpenPGP Message Format”, Network Associates, IN-Root-CA Individual Network e.V., EIS Corporation, November 1998

[6] N. Cottin, O. Baala, J. Gaber, M. Wack, “Management and QoS in Distributed Systems”, *in Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications* (PDPTA'00), Vol. III, Las Vegas, NV, June 2000

[7] H. Dobbertin, A. Bosselaers, B. Preneel, “RIPEMD-160, a strengthened version of RIPEMD”, Fast Software Encryption, LNCS vol. 1039, D. Gollmann Ed., pp. 71-82, 1996

[8] O. Dubuisson, “ASN.1: Communication between Heterogeneous Systems”, Morgan Kaufmann Publishers, ISBN 0-12-6333361-0, June 2000

[9] D. Eastlake, T. Goldstein, “ECML v1.1: Field Specifications for E-Commerce”, Motorola, Brodia, April 2001

[10] M. Elkins, D. Del Torto, R. Levien, T. Roessler, “Draft: MIME Security with OpenPGP”, Network Presence LLC., CryptoRights Foundation, University of California at Berkeley, April 2001

[11] European Telecommunications Standards Institute, *ETSI TS 101 733 v1.2.2*, “Electronic Signature Formats”, December 2000

[12] European Telecommunications Standards Institute, “Policy requirements for time-stamping authorities”, *Draft ETSI TS XXXX STF 178-T1 draft H*, technical specification, ref. DES/SEC-004007-2, Sophia Antipolis, July 2001

[13] J. Farley, *JAVA Distributed Computing*, O'Reilly and Associates, USA, ISBN 1-56592-206-9, January 1998

[14] A. O. Freier, P. Karlton, P. C. Kocher, “Draft: The SSL Protocol Version 3.0”, Netscape Communications, Independant Consultant, November 1996

[15] America Online and the Federal Trade Commission, “Guide to Online Payments”, available on-line at www.ftc.gov/bcp/conline/payments.htm, March 1999

[16] S. Garfinkel, *PGP: Pretty Good Privacy*, First Edition, O'Reilly, ISBN 1-56592-098-8, December 1994

[17] S. Garfinkel, E. H. Spafford, *Web Security & Commerce*, First Edition, O'Reilly, ISBN 1-56592-269-7, July 1997

[18] R. Housley, W. Ford, W. Polk, D. Solo, ‘”RFC 2459: Internet X.509 Public Key Infrastructure, Certificate and CRL Profile”, Spyrus, VeriSign and Citicorp, January 1999

[19] CSOEC-GT, “Guide de l'horodatage sécurisé”, *draft IALTA/SCOEC-GT Horodatage*, IALTA France, July 2001

[20] D. Johnson, A. Menezes, “The Elliptic Curve Digital Signature Algorithm (ECDSA)”, Certicom Research and University of Waterloo, Technical Report CORR 99-34, Dept. of C&O, University of Waterloo, Canada, August 1999

[21] M. Kaeo, *Designing Network Security*, Macmillan Technical Publishing, USA, ISBN 1-57870-043-4, 1999

[22] B. S. Kaliski Jr, “RFC 1319: The MD2 Message-Digest Algorithm”, RSA Laboratories, January 1992

[23] S. Kent, “RFC 1422: Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management”, BBN, IAB IRTF PSRG, IETF PEM WG, February 1993

[24] J. Linn, “RFC 1421: Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures”, IAB IRTF PSRG, IETF PEM WG, February 1993

[25] H. X. Mel, D. Baker, *Cryptography Decrypted*, Pearson Education Corporate, ISBN 0-201-61647-5, 2001

[26] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, USA, ISBN 0-8493-8523-7, February 2001

[27] M. Myers, C. Adams, D. Solo, D. Kemp, “RFC 2511: Internet X.509 Certificate Request Message Format”, VeriSign, Entrust Technologies, Citicorp, DoD, March 1999

[28] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, “RFC 2560: X.509 Internet Public Key Infrastructure, Online Certificate Status Protocol – OCSP”, VeriSign, CertCo, ValiCert, My CFO, Entrust Technologies, June 1999

[29] M. Myers, R. Ankney, C. Adams, S. Farrell, C. Covey, “Online Certificate Status Protocol, version 2”, *draft-ietf-pkix-ocspv2-02*, March 2001

[30] National Institute of Standards and Technology, “Secure Hash Standard (SHS)”, Federal Information Processing Standards Publication, FIPS PUB 180-1, April 1995

[31] National Institute of Standards and Technology, “Electronic Data Interchange (EDI)”, Federal Information Processing Standards Publication, FIPS PUB 161-2, April 1996

[32] National Institute of Standards and Technology, “Data Encryption Standard (DES)”, Federal Information Processing Standards Publication, FIPS PUB 46-3, October 1999

[33] National Institute of Standards and Technology, “Digital Signature Standard (DSS)”, Federal Information Processing Standards Publication, FIPS PUB 186-2, January 2000

[34] J. W. Parsons, “Draft: Electronic Commerce Modeling Language (ECML): Version 2 Specification”, American Express, February 2001

[35] RSA Data Security Inc., “Public Key Cryptography Standards, PKCS 1-12”, available on-line at ftp://ftp.rsa.com/pub/pkcs, 1993

[36] B. Preneel, A. Bosselaers, H. Dobbertin, “The cryptographic hash function RIPEMD-160”, *CryptoBytes*, vol. 3, No. 2, pp. 9-14, 1997

[37] B. Ramsdell, “RFC 2632: S/MIME Version 3 Certificate Handling”, Worldtalk, June 1999

[38] B. Ramsdell, “RFC 2633: S/MIME Version 3 Message Specification”, Worldtalk, June 1999

[39] E. Rescorla, “RFC 2631: Diffie-Hellman Key Agreeement Method”, RTFM Inc., June 1999

[40] R. L. Rivest, “RFC 1320: The MD4 Message-Digest Algorithm”, MIT Laboratory for Computer Science and RSA Data Security, April 1992

[41] R. L. Rivest, “RFC 1321: The MD5 Message-Digest Algorithm”, MIT Laboratory for Computer Science and RSA Data Security Inc., April 1992

[42] Sun Microsystems, “’Lesson: Signing and Verifying JAR Files”, available on-line at http://java.sun.com/docs/tutorial/jar/sign/index.html, 2001

[43] J. M. Ugljesa, “Program Management Reporting Using Electronic Data Interchange”, September 1998