Time-stamping electronic documents and signatures

Nathanael Cottin
MSG Software – SeT laboratory
90000 Belfort, France
nathanael.cottin@msg-software.com

Maxime Wack
UTBM – SeT laboratory
90010 Belfort, France
maxime.wack@utbm.fr

Abdelaziz Sehili
UTBM – SeT laboratory
90010 Belfort, France
abdelaziz.sehili@utbm.fr

Abstract
Time-stamping becomes a vital component of the emerging electronic business infrastructures. The main goal is to provide users with time management and - possibly signed- electronic content protection. We address in this article the purpose and usage of time-stamps on electronic documents and electronic signatures. A discussion is opened on the effective need of time-stamping electronic signatures through the study of two time-stamping technologies.

Keywords
Time-stamp, electronic signature, PKCS, EDCI, TSP, OCSP.

INTRODUCTION
We point out in this article time-stamping juridical and technical needs for electronic documents as well as electronic signatures. We do not take into account widely distributed public documents but only documents having restricted or private access as well as electronic transactions such as teleservices. Yet, a wide distribution of a public document (i.e. proceedings) agrees with ownership needs as many users may have knowledge of their existence. However research non-public ideas, prototypes specifications and many more electronic documents need to be protected. Electronic signature is not necessarily required. Proof of existence and ownership may be achieved provided use of time-stamps.

Given our study's main directions, we enumerate three possible ways to make sure a datum is owned by a party: send the datum (or its digest) to a Storage Authority (SA) [1] or another authority, send the digital signature (or its digest) to a SA, or ask a Time Stamp Authority (TSA) for a time-stamp token.

In all cases, one or more trusted third parties (TTPs) [2], also known as trusted service providers (TSPs) [3], must intervene in the ownership attribution or transaction process. These "neutral" TTPs guarantee that the date of the ownership operation cannot be forged. They act as a third party witnesses that protect and seal electronic contents such as patents or contracts.

This article is organized as follows:

- First part describes time-stamping needs in terms of proof of possession, transaction effectiveness denial and electronic signature validation.
- Second part notices common security and structural issues on time-stamping. A subset of disaster recovery concepts related with time-stamping are introduced at this occasion.
- Third part presents two electronic signature management technologies, PKCS and EDCI, to illustrate different technical approaches for signatures long-term validation, based on time-stamping and public-key infrastructures (PKI) services [4]. This comparison makes evidence of time-stamping usage in electronic signatures.
- We finally conclude by presenting further work-in-progress related with TTPs integration.

TIME-STAMPING OVERVIEW
A time-stamp is a *certified* date and time created upon a given datum. This datum may be any digital content such as a document or an electronic signature. A time-stamp is usually used to give a date and time on a strong collision resistant digested value (footprint) of the content. It makes evidence that the time-stamped datum existed before the time-stamp token was created (see 1) and therefore proves ownership:

- The only way to certify that the content existed during *period 1* (or, at least before any time-stamping process) is to send it (or its digest) to a Storage Authority (SA). As a TTP, this SA is responsible of the content integrity and can therefore provide ownership information. Unfortunately, there is no means of redress in case another party contests ownership until this process (or a time-stamping process) is performed.
- Once a time-stamp has been asserted to the content ($t + \varepsilon$), the evidence of possession can be established *period 2*. It is then much harder for a malicious third party to prove ownership because the procedure implies to prove that the time-stamp is actually a fake.

Yet the shorter time delay ε between content creation date t and time-stamp generation $t + \varepsilon$, the more security and disaster limitation: it is advised that $\varepsilon \to 0^+$ to protect signature validity from revocation, expiration and forgery.

Juridical Considerations
The date and time mentioned within a time-stamp token must be delivered by a trusted third party (TTP) called *Time Stamp Authority* (TSA). This TTP is requested to be in compliance with technical standards and policies in order to be accredited. This accreditation is made official when a trusted certificate authority (CA) delivers a TSA certificate and the TSA is registered as such by governments or other TTPs. The effective juridical implication of time-stamps in

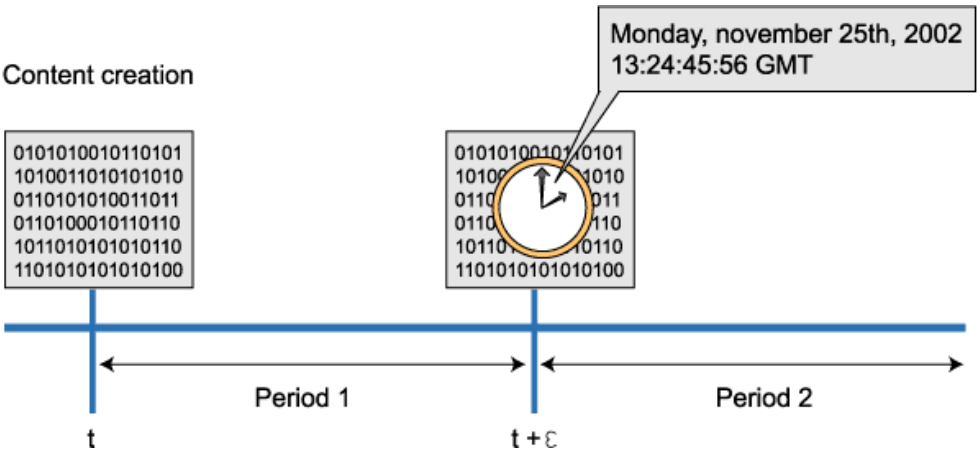

Figure 1. Time-stamp common use

electronic legal documents has not been yet defined in many countries. However, local working groups such as IALTA [5] publish documents that may help jurists define time-stamps and TSAs related laws.

Technical Considerations

A time-stamp should not be compared with a local time on a system, as it is issued and electronically signed by a TTP called Time Stamp Authority (TSA).

It may be used to add electronic signature's long-term verification abilities [6] [7] to fulfill EESSI (European Electronic Signature Standardisation Initiative) recommendations ([8], par. 6.3.1): "*Signature and certificate chain must be time-stamped*". It testifies that an electronic signature was generated before the date and time mentioned in the time-stamp token. This may be a critical information to make sure the signature was created during the signatory's certificate validity period and before any revocation. Without such (or similar) information, verifiers cannot establish a link between the signatory's certificate validity status and the electronic signature.

In the signature process, a TSA's time-stamp freezes the signatory's signature. This time-stamp appears then to be the weakest link in the signature validation chain as it is used for both signatory's signature protection and time control. Indeed, TSAs must be aware of or conform to the latest technologies to make sure their signatures cannot be forged.

COMMON SECURITY AND STRUCTURAL ISSUES

As shown before, time-stamps play a key role for both content proof of anteriority and electronic signature protection and long-term validation. The delivery process implies to interact with one or more TSAs. Time Stamp Protocol (TSP) [9] defines data structures as well as requirements that TSAs must conform with. The basic idea is that TSAs only time-stamp digests (hashcodes) and should not take possession of any significant content.

We mainly indicate and comment a few recommendations. According to TSP, a TSA is required:

- "*to use a trustworthy source of time*": the source that gives the time-stamp value must supply accreditors recommendations, such as NTP and atomic clock.
- "*to examine the OID of the one-way collision resistant hash-function and to verify that the hash value length is consistent with the hash algorithm*": a TSA may reject a time-stamp request in case the digest algorithm used to create the provided hashed value does not conform with its time-stamping policies. This may occur if this algorithm is no more collision resistant.
- "*to sign each time-stamp token using a key generated exclusively for this purpose and have this property of the key indicated on the corresponding certificate*": time-stamps are electronically signed by TSAs using their TSA certificate.

Three main limitations of time-stamps may be put forward:

- TSA's signature validity period.
- TSA signing key compromise.
- Signatory's cautionary period.

Signature Validity Period

We assume that the signature and certificate validity periods refer to a unique period of time. This is currently the case with x.509v3 standard [10] which encloses a single public-key.

Therefore the signature of the TSA on a time-stamp token expires as its certificate validity period ends. This time-stamp must be actualized using one of the following:

- The time-stamp is replaced by a new one which encloses the same date and time. This procedure is not standardized as it may leak security considerations and open trapdoors.
- Instead, the time-stamp can be considered as another content and time-stamped (recursive process). This is the standard procedure.

The main issue of time-stamps validity periods is that each time-stamp will necessary expire and must be actualized. To overcome this problem, we recommend to make use of

SAs and do not consider juridical aspects until the content is not time-stamped and securely stored.

TSA Key Compromise or Loss

A key compromise or loss may eventually occur in case an attacker breaks the TSA's private key protection system (key compromise) or the private key is destroyed (key loss). Unlike PKIs, TSAs cannot handle backup keys to generate doubled time-stamps (first using the "official" keypair and second using backup keys). Yet, whereas CAs are able to revoke all issued certificates in case a disaster occurs and make backup certificates publicly available, this procedure is not practicable with TSAs as there may be too many time-stamps to revoke. Furthermore, there would be no way to make a distinction between a "real" time-stamp (issued by a TSA) and a fake in case the TSA's signature (private) key is discovered.

This is the main reason why standard disaster recovery procedures [9] recommend to ask for more than a single time-stamp. In this case, all implied TSAs' private keys should be compromised at the same time to invalidate all time-stamps on a given content.

We derived this idea to propose a "cross-certification"-like system that relies on accreditors [2]: depending on the requested time-stamping policy, the TSA is able to ask other TSAs or specialized TTPs called *accreditors* to cross-validate the time-stamp token. This process is quite similar to the standard process. However:

- The multiple time-stamping process is delegated to the requested TSA's discretion. Requesters do not need to worry about disaster recovery procedures as they directly receive countersigned time-stamp tokens. The number of required countersignatures and token acceptance delta-times may be specified by TSA's policies.
- A single time-stamp token is necessary instead of many different tokens that indicate different date and time values. It is generated and signed by the requested TSA and given to the accreditors.

A simple possible implementation of this principle would be:

```
TimeStamp ::= SEQUENCE {
  sts      SignedTimeStamp,
  counter Signatures,
}

SignedTimeStamp ::= SEQUENCE {
  value    TimeStampValue,
  sign     Signature
}

TimeStampValue ::= SEQUENCE {
  policy  Policy,
  token   TimeStampToken
  - defined by RFC3161
}
```

Where *counter* indicates that the signed time-stamp is countersigned by at least one accreditor. The signature internal structure may vary depending on the implementation. Effectiveness of this service may suffer from a denial of service (DoS) vulnerability that may lead to invalidate the accreditation process as the delay of *sts* acceptance would be expired.

Signatory's Cautionary Period

Depending on the validation policy applied to a given electronic signature [11], the verifier shall not know the effective status of the signatory's PKC. A cautionary period may thus be required to make sure this PKC was valid at the date and time indicated by the time-stamp token. This is due to the "*inevitable delay between a compromise or loss of key being noted, and a report of revocation being distributed*" [7]. This cautionary period may be mentioned by the verification policy and shall depend on the verification

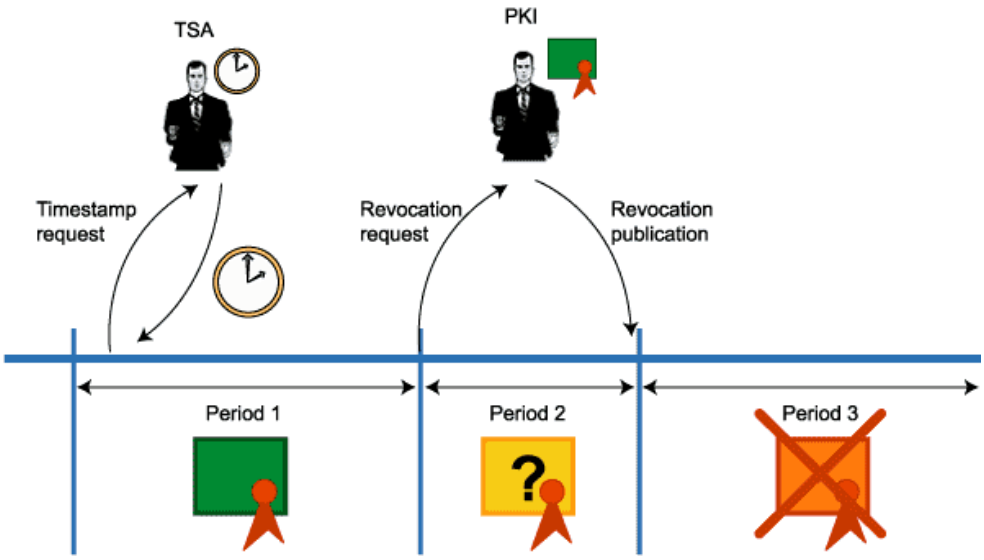

Figure 2. Cautionary period

protocol (use of on-line revocation verifications or off-line CRLs for example).

Figure 2 illustrates this time latency between revocation notification and effective publishing by the PKI. The different periods represent three possible signatory's certificate status values: respectively valid, unknown and revoked. Suspension and expiration status are not taken into account here. In case the verifier is not aware of the signatory's revocation request, it cannot undoubtly establish the validity status of the latter's certificate (*period 2*) and should wait for the cautionary period to be notified of the possible revocation. Therefore we suggest this period to be equal or greater than the CRL (eventually delta-CRL) or OCSP [12] actualization periods given by the PKI's Certificate Practice Statements (CPS).

COMPARISON BETWEEN PKCS AND EDCI

The purpose of this comparison is to point out the effectiveness of time-stamps in the area of electronic signature. We illustrate this objective with two different approaches of time-stamping, namely PKCS and EDCI.

Both solutions rely on x.509 public-key infrastructures (PKIX) services to provide signatories with public-key certificates (PKC) that testify their identity and give certificates validity status via OCSP responders or CRLs.

This study deals with major off-line and on-line long-term signature creation and validation scenarios. We only describe on-line procedures that imply active connections with TTPs and refer to off-line solutions as alternative ways.

Overviews

PKCS Overview

The Public Key Cryptography Standards are RSA Data Security Inc.'s attempt to provide an industry standard interface for public cryptography to acheive data protection and authentication. It is a private collection of standards developed working with a variety of companies. "*These standards cover RSA encryption, Diffie-Hellman key agreement, password-based encryption, extended-certificate syntax, cryptographic message syntax, private-key information syntax, and certification request syntax, as well as selected attributes*" [13].

We focus on PKCS#7 [14], PKCS#9 [15] and related standards that describe envelopes for signed messages (contents). PKCS#7 provides a way to cipher and sign data. CRL can optionally be appended to the list of signatories to allow digital signature validation. It is based on different content types (e.g. *signed-data*, *signed-and-enveloped-data* or *encrypted-data*).

We exclusively discuss *signed-data* bags to estimate the importance of time-stamps on signatures validation. A signed-data component represents signatory's information. This information is structured as follows:

```
SignedData ::= SEQUENCE {
  version       Version,
  digestAlgorithms
    DigestAlgorithmIdentifiers,
  contentInfo   ContentInfo,
  certificates
    [0] IMPLICIT
    ExtendedCertificatesAndCertificates
    OPTIONAL,
  crls
    [1] IMPLICIT
    CertificateRevocationLists
    OPTIONAL,
  signerInfos   SignerInfos
}

SignerInfos ::= SET OF SignerInfo

SignerInfo ::= SEQUENCE {
  version       Version,
  issuerAndSerialNumber
    IssuerAndSerialNumber,
  digestAlgorithm
    DigestAlgorithmIdentifier,
  authenticatedAttributes
    [0] IMPLICIT
    Attributes OPTIONAL,
  digestEncryptionAlgorithm
    DigestEncryptionAlgorithmIdentifier,
  encryptedDigest
    EncryptedDigest,
  unauthenticatedAttributes
    [1] IMPLICIT
    Attributes OPTIONAL
}
```

The signing time must be enclosed as an authenticated attribute. PKCS#9 authenticated attributes are signed (i.e., authenticated) by the signatory along with the content. According to PKCS#9, signing time "*specifies the time at which the signer (purportedly) performed the signing process*". Nevertheless, "*no requirement is imposed concerning the correctness of the signing time, and acceptance of a purported signing time is a matter of a recipient's discretion. It is expected, however, that some signers, such as time-stamp servers, will be trusted implicitly*".

As stated, no time-stamp integration is performed by PKCS standards. Thus, we suggest to enclose time-stamps as unauthenticated attributes. PKCS wrapped time-stamps may be identified by *pkcs-9-at-timeStamp* and expressed with the object identifier {*pkcs-9-at-signingTime 1*}. The standard internal structure of a time-stamp token is described by TSP. The time-stamped message is necessarily a *MessageImprint* defined as follows:

```
MessageImprint ::= SEQUENCE  {
  hashAlgorithm      AlgorithmIdentifier,
  hashedMessage      OCTET STRING
}
```

This means that requesters do not need to present significant contents but digested values created using one-way, collision resistant digest algorithms. The digest algorithm identifier is also provided and time-stamped for source content validation purpose.

EDCI Overview

Electronic Data Certification Infrastructure (EDCI) is a general framework introduced by Trustronic [16]. This distributed architecture relies on multiple TTPs and especially Certificate and Time Stamp Authorities.

The main idea of EDCI is that time-stamping is not required for digital signature long-term verification and validation. Instead, a certificate of validity (COV) may be used and appended to the electronic signature, as shown by EDCI signature's internal structure (*status* field in particular):

```
Signature ::= SEQUENCE {
  sign        Sign,
  ts      [0] TimeStamps OPTIONAL,
  status  [1] CertifiedSignatureValidity
              OPTIONAL,
  counter [2] Signatures OPTIONAL,
  extns   [3] Extensions OPTIONAL
}

Sign ::= SEQUENCE {
  signer      ExtendedCertificate,
  value       MessageImprint,
  - as defined by RFC3161
  extns       Extensions OPTIONAL
}

CertifiedSignatureValidity ::= SEQUENCE {
  validity    SignatureValidity,
  sign        Signature
}

SignatureValidity ::= SEQUENCE {
  sign        MessageImprint,
  - digest of the requester's signature
  status      ValidityStatus,
  ts          GeneralizedTime OPTIONAL
}

ValidityStatus ::= PKIStatus
- defined in RFC3161
```

Where :

- *Sign* refers to a signature. It basically encloses a PKC and a signature value (encrypted digested value).
- *TimeStamps* is a list of time-stamps for standardized time-stamps usage conformance.
- *CertifiedSignatureValidity* is a certificate of validity (COV). It is given by a Signature Validation Responder (SVR) in response to a COV request and includes the digested value of the provided signature.
- *Signatures* is a sequence of signatures (recursive definition).
- *Extensions* refer to x.509 standard extensions.

As for TSP, EDCI time value is expressed as a *GeneralizedTime*:

```
TimeStampToken ::= SEQUENCE {
  hash          MessageImprint,
  value         GeneralizedTime
}
```

This token is to be enclosed within a *TimeStamp* structure, along with the TSA's signature that authenticates the token.

Time-Stamped Electronic Signature Creation Scenarios

PKCS Electronic Signature Creation

An electronic signature is created on a content info using the *signed-data* and *signed-and-enveloped-data* content types. We do not consider the degenerated case where signed data are actually not signed (used for PKCs and CRLs distribution). PKCS allows to create multiple signatures on a given content. The above-mentioned *SignedData* structure is able to handle co-signatures (signatures created at the same level). Counter-signatures are provided by recursively sign another signed content.

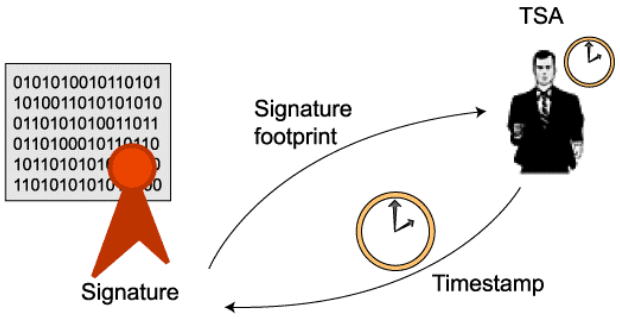

Figure 3. Standard long-term signature creation

Signature creation may integrate time-stamps (see Figure 3) for each signer at a given level [17] to achieve signature long-term validation. However this case is not specifically described in PKCS standards (#7 in particular) but rather in [6]. Such signatures must conform to ES, ES-T, ES-C and eventually ES-X structures (see Figure 4) that make use of time-stamps at different levels (ES-T and ES-X):

- *ES* (Electronic Signature) contains the minimum elements that a signer must provide.
- *ES-T* (Electronic Signature with Time-stamp) is an enhanced version of ES where the signature value is time-stamped (first time-stamp – level 1).
- *ES-C* (Electronic Signature with Complete validation data) basically wraps an *ES-T* to present the digital signature in a form that may not be repudiable by providing certificates status validation means such as CRLs or OCSP references.
- *ES-X* (Electronic Signature eXtended) applies a time-stamp over *ES-C* data to freeze certificate and signature-related information with currently available cryptography (second time-stamp – level 2).

The time-stamping process (primarily in PKCS to create an ES-T-like bag) may be performed by each signatory (or before the current bag is nested within another bag) to obtain a date and time as close to the signature creation as possible. A delta-time indicator can be computed using both time-stamps and an authenticated attribute of the signature called *signing time*. The time-stamp may be rejected in case this indicator value exceeds the maximum value according to the applied policy.

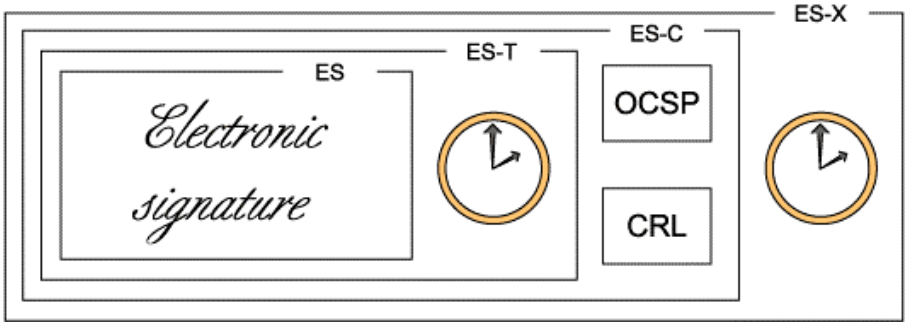

Figure 4. Standard long-term signature inner data structures

Time-stamps are requested to TSAs. Interactions between TSAs and signature software are described by TSP. Each TSA is requested to generate time-stamp tokens that enclose the digest (expressed as a *MessageImprint*) of the to-be-time-stamped message (TBT message) as well as a time value (preferably expressed in Zulu generalized time). This token is then signed using a specific certificate only designed for this purpose.

Each received time-stamp may be integrated within the signed-data bag as an unauthenticated attribute. It is not required to make time-stamps authenticated attributed for two main reasons:

- A time-stamp is self-authenticated: no authentication means other than OCSP or CRL check is necessary.
- A time-stamp is created once the digital signature has been created. It is built upon the signature's digested value (*MessageImprint*).

Conforming to our proposed time-stamping procedure, a primary ("master") TSA is ordered to generate a single time-stamp that may be accredited (countersigned) by secondary TSAs (accreditors).

EDCI Electronic Signature Creation

EDCI allows to create signatures that handle short-term and long-term validation. The long-term validation ability lays on the introduction of a COV. This certificate is signed by a PKI-related component such as a CA service or another trusted responder. It is created using a signatory's electronic signature.

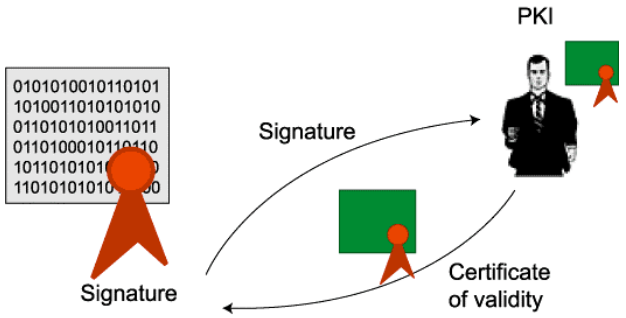

Figure 5. EDCI long-term signature creation

This PKI service, depicted by Figure 5 and called Signature Validation Service (SVS), returns similar certificate validity status information as an OCSP responder would do. However, this information is to be enclosed within the electronic signature as a *CertifiedSignatureValidity* value.

Depending on the requested generation policy, the SVR may add a better security level to the COV by asking either multiple TSAs or a primary TSA for time-stamping its signature (see Figure 6). Standard security and disaster recovery rules then apply to the SVR's signature on the COV. Time-stamping operations also provide means for proof of content possession as time-stamps transitively authenticate and give date and time information about the original signed content.

Time-Stamped Electronic Signature Validation Scenarios

PKCS Electronic Signature Validation

The standard way to validate an electronic signature designed for short-term verification is to check that the digest created from the original signed content matches the signature. This is performed as follows (see Figure 7):

1. A new digest (whitness) is generated by the verifier using a similar digest algorithm as the algorithm used by the signatory during the signature creation process.
2. A second digest (original) is extracted from the electronic signature using the signatory's PKC.
3. Both digests must be equivalent (bit-per-bit comparison) to authenticate the original (unsigned) content. The signature is otherwise rejected.
4. The validity of the signatory's PKC is determined using the currently available CRL or via an OCSP responder using appended time-stamps.

In case a set of time-stamps is also given by the signatory. These time-stamps may either be enclosed within the signed content or provided by other means (detached time-stamps). In such a case, signatures may be ready for long-term validation.

Local CRLs (or CRLs provided by the signed content) may then be used to also determine the TSA's signature validity to accept or reject the time-stamp token. Once at least one time-stamp token is accepted, the signatory's signature status of validity can by obtained by either CRL check or using OCSP. In case the signatory's PKC hass been revoked, the OCSP response encloses the date of revocation. The interpretation of the CRL or OCSP response is left to the verifier's discretion. The latter may then accept the sig-

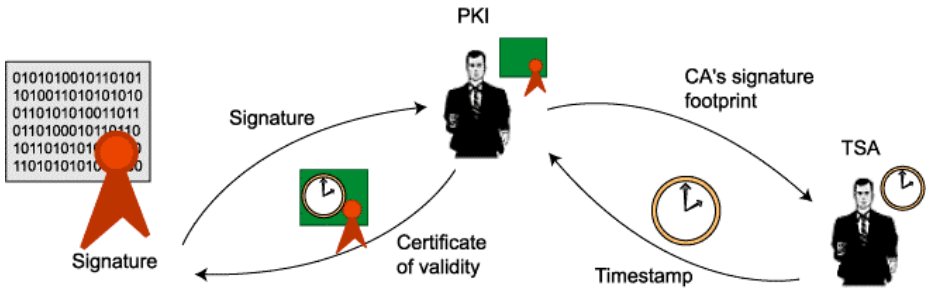

Figure 6. EDCI long-term signature creation with time-stamping

nature of the signatory if the time-stamp token indicates a date and time prior to the instant of revocation and depending on the cautionary period.

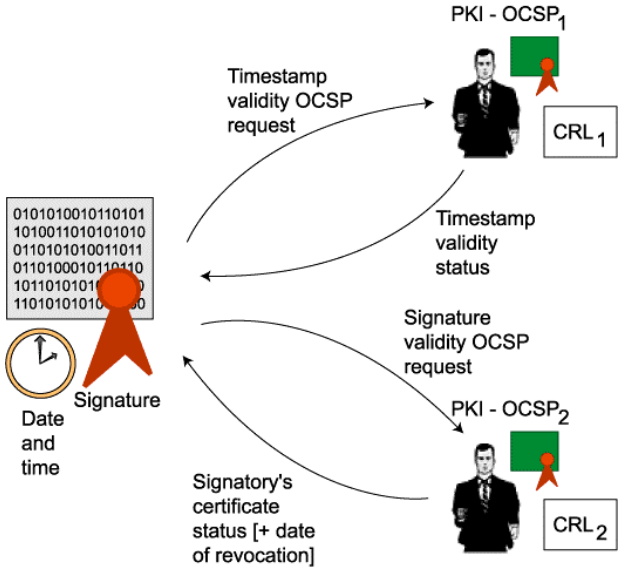

Figure 7. Standard long-term signature validation

As standard procedures recommend to ask for at least two time-stamps, the question of time-stamp selection for signature validation arises. We suggest to order the sequence of time-stamps on the time-stamp tokens values. The first time-stamp used for signature validation therefore indicates the lowest time value. This selection limits signature status misinterpretation as the signatory's certificate may be revoked (or eventually expire) between two time-stamps deliveries.

Not only our time-stamp accreditation proposal exempts end-user applications from multiple TSAs connections, it also prevents from signature validity controversy as a single time value is considered during the validation process.

EDCI Electronic Signature Validation

The validation process depends on the ability for the verifier to accept or reject the COV. This verification may be performed using off-line local Authorities Revocation Lists (ARLs). The main interest of an ARL is that it only lists TTPs revoked certificates, contrary to CRLs that enclose end users' certificates. Authorities private keys and certificates environment conform to global security recommendations and rules that do not compare with common signatories': TTPs own strengthened keypairs, need physical private key protection, etc. Consequently, ARLs information should not need to be regularly updated but only when notified by TTPs (email and web notification or other means such as newspaper, radio and television).

Once the electronic signature holds a COV, there should be no need of on-line connection to any TTP and local ARL information should be sufficient to determine the validity of the COV and thus the signatory's signature status of validity. The following validation steps may be required:

1. Generation of a digest from the original (unsigned) content and comparison with the signature data (same procedure as previously described).
2. Acceptance of the certificate of validity using either an ARL or an on-line connection to a SVR. In case the certificate is time-stamped, an extra connection may be performed.
3. Validation of the signatory's PKC. The status of the signature (and therefore of the signatory's certificate) is given by the certificate of validity *ValidityStatus* field.

Hence, verifiers may ask a SVR to get the current CA's signing certificate status to make sure it has not been revoked (see Figure 8).

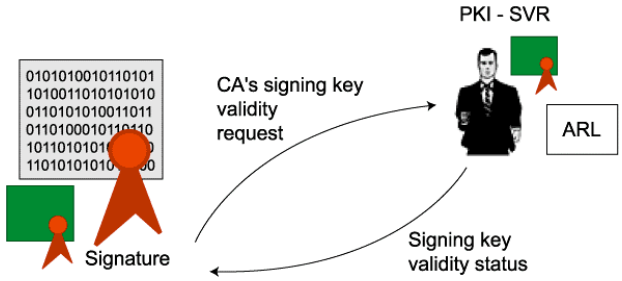

Figure 8. EDCI long-term signature validation

In case the COV is time-stamped, it is possible to determine whether the CA's signing certificate is valid, suspended or revoked at the time the COV was created. If not timestmaped, it is required that the SVS includes its local time during the COV generation process so that its certificate can be verified upon the ARL validity period. This means that time-stamping COVs is not necessary required to attribute electronic signatures long-term validation capabili-

ties. The time-stamp validation procedure may also implement our accreditation proposal as described before.

CONCLUSION

We outlined both juridical and technical implications of time-stamping electronic contents.

We suggested to introduce accreditors to countersign time-stamp tokens and prevent from disaster recovery without requiring multiple TSAs connections from signature applications. The underlined protocol is based on a primary TSA that generates and signs a single time-stamp token. Its signature is then countersigned by secondary TSAs called "accreditors".

Focusing on PKCS and EDCI time-stamping approaches, we determined whether time-stamping is a necessary process to provide electronic signatures with long-term verification or not. It appears that a time value is not required to validate an electronic signature as a certified status of validity is sufficient. We demonstrated the interest of EDCI's certificates of validity (COV) that directly assign certificates status on electronic signatures. Such certified signatures do not require further on-line connections or CRLs check to assign a status of validity. Only ARLs off-line consultations (or similar on-line requests) may be performed to make sure that PKI's signing certificates have not been revoked. These COVs may be introduced in PKCS standards as unauthenticated attributes and thus provide PKCS with COV capabilities.

We also introduced the notion of PKI's Signature Validation Service (SVS). This service is to be used to generate certificates of validity and is a PKI extra component similar to an extended OCSP responder called Signature Validation Responder (SVR).

To conclude this study, security concerns on certificates used by TTPs reveal that time-stamping or other means that provide signatures with long-term validation abilities suffer from their necessary renewal. This occurs when the TTPs appended signatures reach their end of validity periods. Thus we suggest to consider having recourse to storage authorities (SAs) and electronic notaries to store signed contents (or detached signatures). This would first avoid renewal considerations but also make sure that the signatures on the stored content do not imitate (mimic) "old" signatures created with new technology. An alternative solution would be to contact SVRs for COV signature renewal.

Our further work on time-stamping is to define policies related with time-stamps accreditation protocols and integrate other TTPs presented by EDCI within our signed contents management framework under development. We may then discuss XML signatures [18] that include time-stamping as a signed information.

REFERENCES

[1] Groupe de Travail Commun Archivage, CSOEC, IALTA france, EDIFICAS, *Guide de l'archivage électronique sécurisé*, July 2000

[2] N. Cottin, B. Mignot, M. Wack, *Authentication and enterprise secured data storage*, proc. of the 16th International Conference on Emerging Technologies on Factory Automation (ETFA'01), Antibes Juan-les-Pins, France, October 2002

[3] J. Ross, D. Pinkas, N. Pope, *Electronic Signature Policies*, RFC3125, September 2001

[4] C. Adams, D. Farrell, *Internet X.509 Public Key Infrastructure Certificate Management Protocols*, RFC2510, March 1999

[5] Groupe de Travail Commun Horodatage, CSOEC, IALTA france, EDIFICAS, *Recommandations pour l'horodatage électronique*, to appear

[6] D. Pinkas, J. Ross, N. Pope, *Electronic Signature Formats for long term electronic signatures*, RFC3126, September 2001

[7] European Telecommunications Standards Institute (ETSI), *Electronic Signatures and Infrastructures (ESI); Electronic Signature Formats*, ETSI TS 101 733 V1.4.0, technical specification, September 2002

[8] European Electronic Signature Standardization Initiative (EESSI), *Final Report of the EESSI Expert Team*, July 1999

[9] C. Adams, P. Cain, D. Pinkas, R. Zuccherato, *Internet X.509 Public Key Infrastructure: Time Stamp Protocol (TSP)*, RFC3161, August 2001

[10] R. Housley, W. Polk, W. Ford, D. Solo, *Internet X.509 Public Key Infrastructure, Certificate and Certificate Revocation List (CRL) Profile*, RFC3280, April 2002

[11] D. Pinkas, *Certificate Validation Protocol*, Internet Draft, October 2002, available on-line at *www.ietf.org*

[12] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, *X.509 Internet Public Key Infrastructure, Online Certificate Status Protocol – OCSP*, RFC2560, June 1999

[13] Burton S. Kaliski Jr., *An Overview of the PKCS Standards*, RSA Laboratories Technical Note, Revised November 1, 1993

[14] Burton S. Kaliski, *Cryptographic Message Syntax*, PKCS#7, version 1.5, March 1998

[15] RSA Laboratories, *Selected Object Classes and Attribute Types*, PKCS#9, version 2.0, February 2000

[16] Trustronic, *EDCI: ASN.1 data structures*, version 1.2b, December 2002, available on-line at *www.trustronic.org*

[17] D. Rieupet, N. Cottin, *Scénarios d'apposition de multiples signatures*, version 1.0, technical document, January 2003

[18] D. Eastlake 3rd, J. Reagle, D. Solo, *(Extensible Markup Language) XML-Signature Syntax and Processing*, RFC3275, March 2002