



NATIONAL SECURITY AUTHORITY

**Version 1.4**

# **Certificate Path Validation**

**19<sup>th</sup> November 2006**

---

**NATIONAL SECURITY AUTHORITY**

Department of Information Security and Electronic Signature

Budatínska č. 30, 850 07 Bratislava 57

<http://www.nbusr.sk/>

E-mail: [sep@nbusr.sk](mailto:sep@nbusr.sk)

# Content

<b>1</b>	<b>Introduction.....</b>	<b>4</b>
<b>2</b>	<b>Scope.....</b>	<b>4</b>
<b>3</b>	<b>References.....</b>	<b>5</b>
<b>4</b>	<b>Abbreviations .....</b>	<b>6</b>
<b>5</b>	<b>Components of the Certification Path.....</b>	<b>7</b>
<b>6</b>	<b>Rules determining the Certification Path Validation .....</b>	<b>10</b>
<b>7</b>	<b>Shift of a control time in signatures verification in the certification path.....</b>	<b>11</b>
<b>8</b>	<b>Archiving and verification of archival certification path.....</b>	<b>12</b>
<b>9</b>	<b>Attributes of the certificate X.509 in the Certification Path Validation.....</b>	<b>13</b>
9.1	Sequence of verified items in the Certification Path Building.....	13
<b>10</b>	<b>Certificate Path Validation .....</b>	<b>14</b>
10.1	The algorithm for Certification Path Building.....	15
10.2	Searching for certification paths through the recursive procedure .....	17
10.3	Algorithm for Certificate Path Validation .....	18
<b>Annex A (informative) Verification on the basis of OCSP with positive response and with OCSP response in accordance with RFC 2560.....</b>		<b>20</b>
A.1	OCSP with positive response on the basis of data from database.....	20
A.2	OCSP response in accordance with RFC 2560 .....	21
<b>Annex B (normative) Verification of a certificate validity.....</b>		<b>22</b>
B.1	Verification on the basis of OCSP .....	22
Table 1	Verification with OCSP .....	22
B.2	Verification on the basis of CRL.....	23
Table 2	Verification with CRL .....	23
<b>Annex C (informative) Revisions made since previous version.....</b>		<b>25</b>
C.1	Additional requirements .....	25
C.2	Updated requirements.....	25
C.3	Clarifications .....	25
C.4	Editorial.....	25
<b>Annex D (informative) The list of used literature.....</b>		<b>26</b>
<b>Annex E History .....</b>		<b>27</b>

## 1 Introduction

The usage of technologies based on PKI and certificates X.509 has obtained a new legal dimension after implementation of requirements from the European Directive 1999/93/ES in the Member States of the European Union. The identical method for application of rules for Certification Path Building and Certification Path Validation is one of the basic conditions to interpret the results, obtained on the basis of technical methods, correctly and deduce right decisions from them. The usage of PKI technology and X.509 certificates allows a great variability. Therefore it is necessary to integrate this variability by definition and publication of profiles for ensuring of unambiguous methods and requirements which will ensure the same results not only within one state but also internationally.

## 2 Scope

The purpose of the document "Certificate Path Validation" is the description of the certification path building and validation process in accordance with the valid legislation of the Slovak Republic, standards X.509, ETSI and RFC requirements for qualified certificates and Qualified Electronic Signatures in an effort to ensure the relevant verification in the conditions of the Slovak Republic and also with respect to achieve an interoperability with the Member States of the European Union.

The interoperability in the countries of the European Union is built on the basis of the European Directive 1999/93/ES and accepting of standardization institutions documents in which the Member States shall ensure that Qualified Electronic Signature in accordance with the Article 5 of the European Directive 1999/93/ES:

- a) satisfies the same legal requirements of the signature in relation to data in electronic form in the same manner as a hand-written signature satisfies those requirements in relation to paper-based data,
- b) is admissible as evidence in legal proceedings.

The editor's effort is not to create the standard itself but to create an unambiguous minimal profile and recommendations for Certification Service Providers, application creators and also electronic signature users.

### 3 References

References to documents defining used types and methods.

[1]	RFC 3280	X.509 PKI Certificate and Certificate Revocation List	04/2002
[2]	RFC 2560	X.509 PKI Online Certificate Status Protocol	06/1999
[3]	RFC 3852	Cryptographic Message Syntax	07/2004
[4]	RFC 3161	Time-Stamp Protocol (TSP)	08/2001
[5]	RFC 3739	Qualified Certificates Profile	03/2004
[6]	ETSI	TS 101 862 V1.3.2 Qualified Certificate Profile	06/2004
[7]	ETSI	TS 101 733 Electronic Signature Formats	
[8]	ETSI	TR 102 272 ASN.1 format for signature policies	
[9]	ETSI	TR 102 038 XML format for signature policies	
[10]	ISO/IEC	ITU-T X.509 / ISO/IEC 9594	08/2005
[11]	ETSI	TS 102 280 X.509 V.3 Cert. Profile for Cert. Issued to Natural Persons	
[12]	ETSI	TS 102 231 Provision of harmonized Trust-service status information	

## 4 Abbreviations

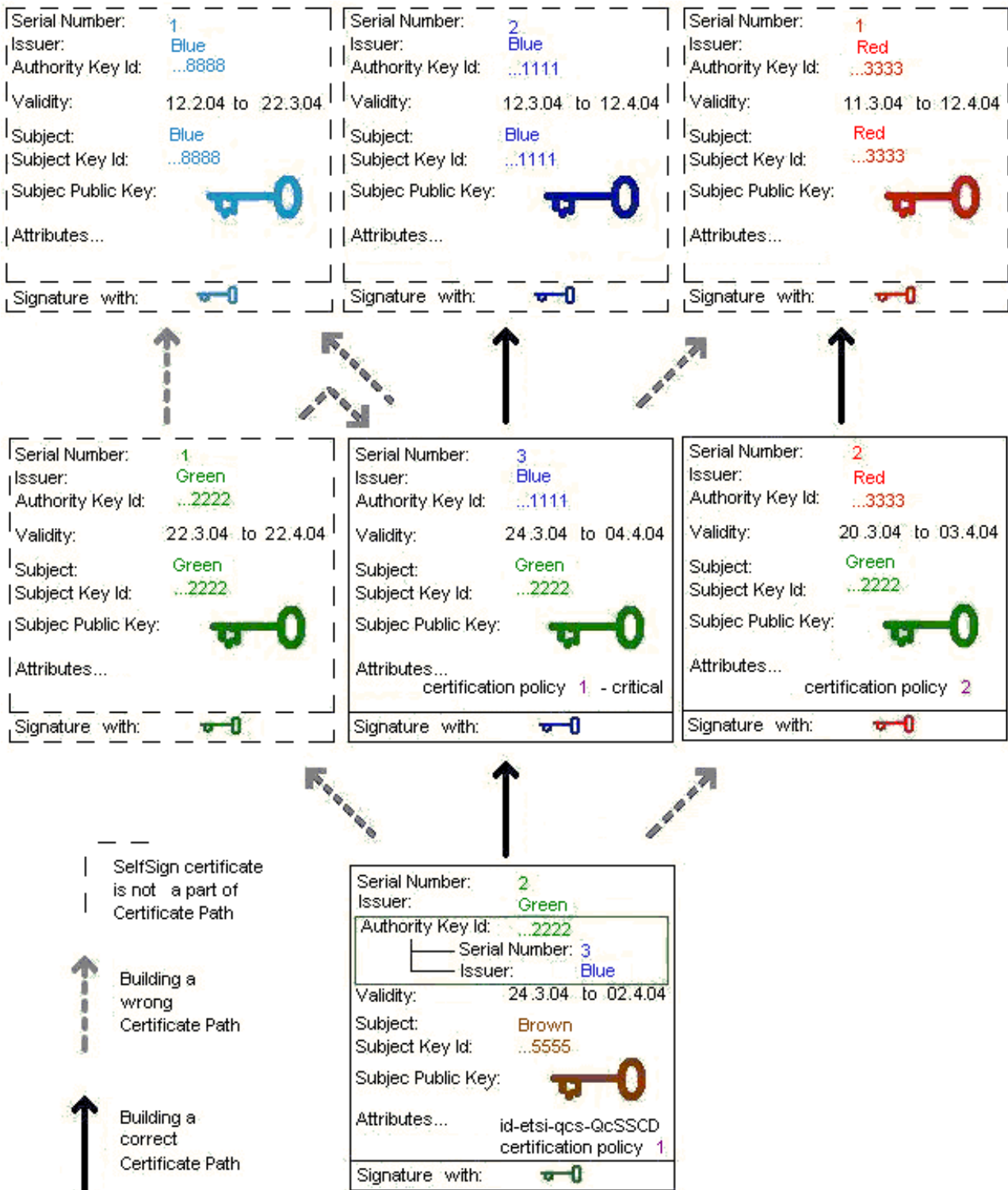
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation 1
CA	Certification Authority
CAeS	CMS Advanced Electronic Signature
CMS	Cryptographic Message Syntax
CRL	Certificate Revocation List
DER	Distinguished Encoding Rules (for ASN.1)
ESS	Enhanced Security Services (enhances CMS)
GMT	Greenwich Mean Time
HTTP	HyperText Transfer Protocol
ISIS-MTT	Industrial Signature Interoperability Standard - MailTrusT
ISO	International Organization for Standardization
MIME	Multipurpose Internet Mail Extensions
OCSP	Online Certificate Status Provider / Protocol
OID	Object Identifier
PKCS	Public Key Cryptographic Standards, Standards published by RSA, Labs.
PKIX	Internet X.509 Public Key Infrastructure
PVM	Path Validation Module
QC	Qualified Certificate
RSA	Rivest, Shamir and Adleman Algorithm
SHA	Secure Hash Algorithm
SSCD	Secure-Signature-Creation Device
TSA	Time-Stamping Authorities
TSP	Time Stamp Protocol
TST	Time-Stamp Token
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XAdES	XML Advanced Electronic Signature
XML	extensible Markup Language
ZEP	Qualified Electronic Signature ( <i>Zaručený elektronický podpis</i> )

## 5 Components of the Certification Path

A Certification Path is being built from **end entity certificate** up to **trusted root CA**. Thus certification paths consist of **end entity certificate** [11] and **certification authorities' certificates** up to CA whose certificate is issued by trusted anchor (the pair of **Distinguished Name** and **Public Key** which is usually presented in the form of the root certificate) **The root certificate (*selfSigned*)** is not a part of the certification path. It is determined only for saving of the trusted anchor (Distinguished Name and Public Key) of **the root CA** in PKIX.

Certification paths built correctly and incorrectly where the certificate is identified by pairs (Certificate Issuer Name, Certificate Serial Number) are specified in the following picture.

1. ***selfSigned* certificates** of certification authorities: an expired certificate, a new certificate which was issued before expiration of the certificate (blue owner - subject and blue key pair) and a certificate of untrusted CA (unknown for a verifier) - (red owner and key pair)
2. ***selfSigned* certificate**, a **certificate of an accredited CA** and CA certificate issued by untrusted CA -the verifier does not trust red *selfSign* certificate (green owner and key pair)
3. **end entity certificate** –of a signer [11] (brown certificate and key pair)



Certification paths that are built incorrectly:

1. (Green, 2) – (Green, 1):
  - a. The root certificates are not a part of the certification path.
  - b. Name in *AuthorityKeyIdentifier authorityCertIssuer* is not the same as *Issuer* name in the issuer certificate, „Blue“ ≠ „Green“
  - c. Certificate serial number in *AuthorityKeyIdentifier authorityCertSerialNumber* is not the same as *serialNumber* in the issuer certificate 3 ≠ 1
2. (Green, 2) – (Red, 2):
  - a. Name in *AuthorityKeyIdentifier authorityCertIssuer* is not the same as *Issuer* name in the issuer certificate „Blue“ ≠ „Red“

- b. Certificate serial number in *AuthorityKeyIdentifier authorityCertSerialNumber* is not the same as *serialNumber* in the issuer certificate,  $3 \neq 2$
3. (Green, 1) – (Blue, 1):
  - a. The root certificates are not a part of the certification path.
  - b. Name in *Issuer* name is not the same as *Subject* name in the issuer certificate, „Green“  $\neq$  „Blue“
  - c. *AuthorityKeyIdentifier keyIdentifier* is not the same as *SubjectKeyIdentifier* in the issuer certificate, ...2222  $\neq$  ...8888
  - d. Certificate signature verification will not be possible to verify with the public key of an incorrect issuer.
  - e. The issued certificate is not issued in certificate validity period of the incorrect issuer.
4. (Blue, 3) – (Blue, 1):
  - a. The root certificates are not a part of the certification path.
  - b. *AuthorityKeyIdentifier keyIdentifier* is not the same as *SubjectKeyIdentifier* in the issuer certificate, ...1111  $\neq$  ...8888
  - c. Certificate signature verification will not be possible to verify with the public key of an incorrect issuer.
  - d. The issued certificate is not issued in certificate validity period of the incorrect issuer.
5. (Blue, 3) – (Red, 1):
  - a. The root certificates are not a part of the certification path.
  - b. Name in *Issuer* name is not the same as *Subject* name in the issuer certificate, „Blue“  $\neq$  „Red“.
  - c. *AuthorityKeyIdentifier keyIdentifier* is not the same as *SubjectKeyIdentifier* in the issuer certificate, ...1111  $\neq$  ...3333
  - d. Certificate signature verification will not be possible to verify with the public key of the incorrect issuer.

## 6 Rules determining the Certification Path Validation

**Control Time** is the time to which electronic signature validity of a document, a certificate, CRL or OCSP in the certification path is verified:

- it is the time of the oldest valid Time Stamp from digital signature which is in the electronic signature;
- in the certificate the time must be from the certificate period (*notBefore*, *notAfter*);
- in CRL it is the time (*thisUpdate*);
- in OCSP it is the time (*producedAt*);
- or it is the time of verification from secure audit record containing a hash of the digital signature (in the signature without the Time Stamp);
- or it is the time which is very close to the current verification time in which CRL (OCSP) was issued for verification of a signer's certificate. It is necessary to obtain CRL (OCSP) for verification of the whole certification path up to the certificate issued by the root CA where each CRL (OCSP) in the certification path, **from** CRL (OCSP) used for document signer's certificate verification **to** the root CA, is issued in the same time or later time then the previous CRL (OCSP).

**CautionPeriod (grace period)** is the period of time that permits the information about the certificate revocation to be published through the revocation process to relying parties; thus it is the minimum time period from control time while verifier has to wait to allow any authorized entity to request a certificate revocation and the relevant revocation status provider to publish revocation status.

Thus *cautionPeriod* is the period of time which is the maximum from *grace period* of individual certification authorities whose certificates build a certification path. Maximum interval for *grace period* is 24 hours interval in accordance with the legislation of the Slovak Republic.

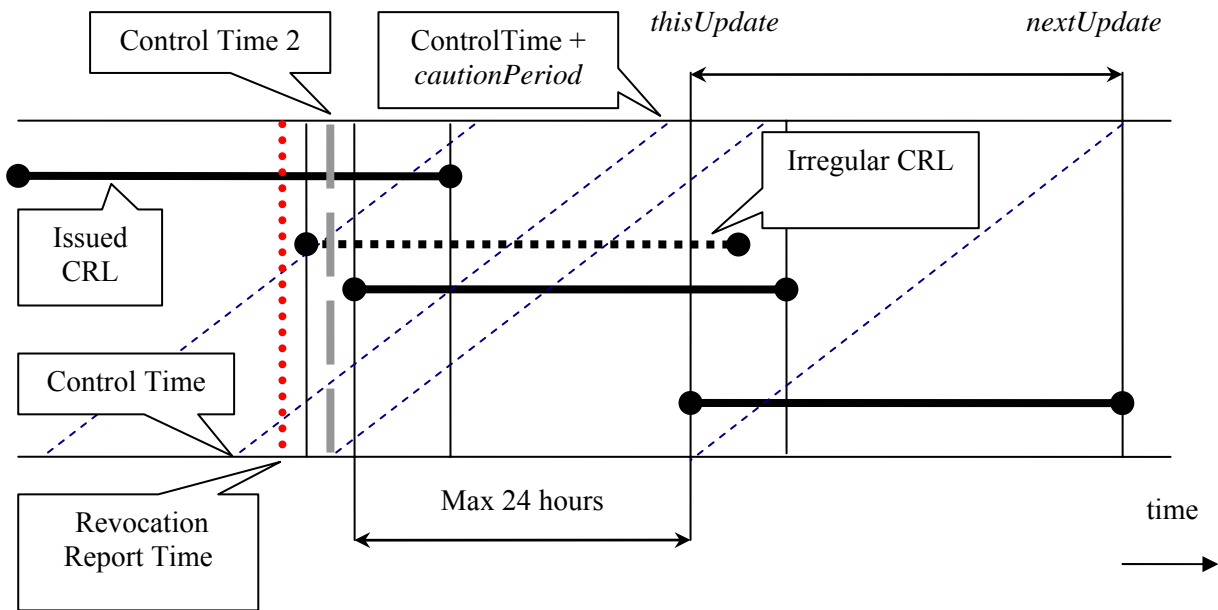
The item *cautionPeriod* can be found for example in the signature policy or in applications settings which verify the certification path.

Time in the *CRL.nextUpdate* can be earlier than time  $CRL.thisUpdate + gracePeriod$  and *CRL.nextUpdate* does not have to be regular in the individual CRL (e. g. in 8 hours, in 16 hours).

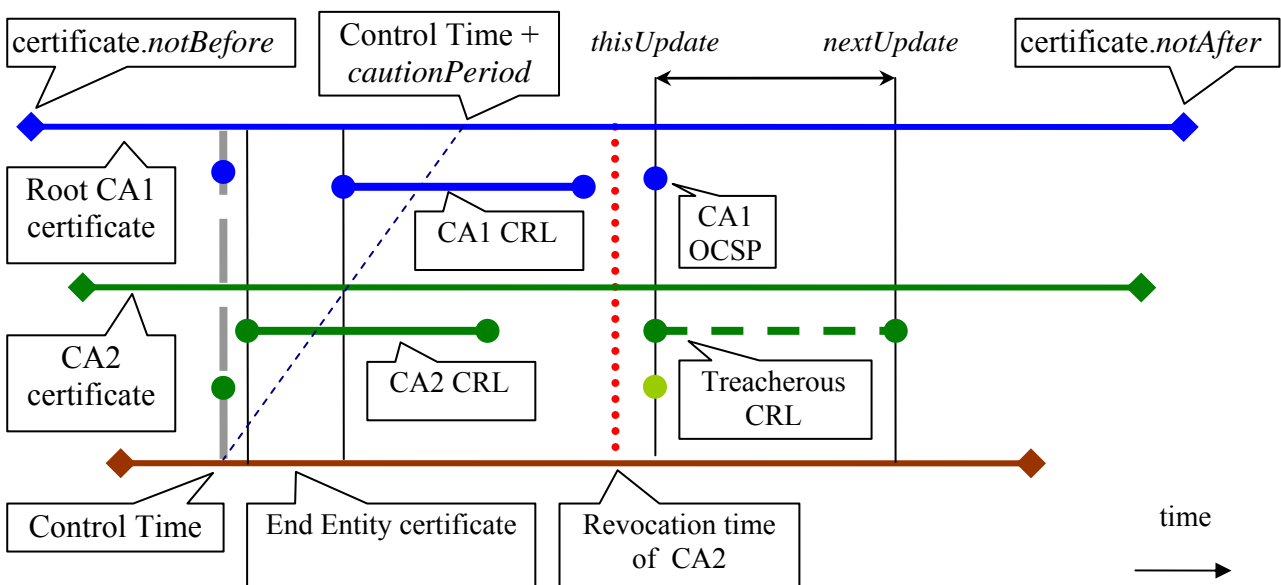
Following sections must be fulfilled in the verification of certificates validity which were issued in accordance with the Slovak legislation (thus they contain OID of the certification policy with the value 1.3.158.36061701.0.0.0.1.2.2):

- a certificate in CRL (OCSP) must not be found in status *certificateHold* and *removeFromCRL*, thus its validity can not be stopped for certain period;
- CRL must not be issued after time *nextUpdate* of the last issued CRL;
- if CRL is issued by CA during the validity of other (currently valid) CRL then in *nextUpdate* of the new CRL the time must be later or equal to time in *nextUpdate* of the latest valid CRL. It means that issued CRL must create a time chain.
- if  $CRL_1(OCSP_1)$  is issued then it is not possible to issue such  $CRL_2(OCSP_2)$  which would revoke a certificate before issuing time of  $CRL_1(OCSP_1)$  in which the certificate has not been revoked yet.

## 7 Shift of a control time in signatures verification in the certification path



In signature verification of CRL (OCSP) with the certificate from the certification path and subsequent verification of the certificate validity by which CRL (OCSP) is verified, it is necessary to fulfill a rule for the choice of the CRL (OCSP) in accordance with issuance time of the CRL.*thisUpdate* (OCSP.*producedAt*). The rule for the choice of CRL (OCSP) is: **for the certificates verification in the certification path, CRL (OCSP) with the issuance time CRL.*thisUpdate* (OCSP.*producedAt*) is being chosen in the way that every CRL (OCSP) of the superior CA is issued after the issuance time CRL.*thisUpdate* (OCSP.*producedAt*) of the subordinate CA.** This rule is necessary to be fulfilled in verification of CRL (OCSP) signature validity that is issued later than a control time because CA certificate which verifies issued CRL (OCSP) could have become revoked later after a control time by hierarchically superior CA.



## 8 Archiving and verification of archival certification path

For protection of all signatures validity that were put into practice by a signer and CA (certificates, CRL, etc.) and might be considered as invalid in the future (due to certificate revocation or an algorithm becomes weak) it is necessary to use a Time Stamp for the whole certification path together with CRL or OCSP.

The whole certification path together with CRL or OCSP can be used for a long-term verification of signatures validity because of the usage of the Time Stamp. This time stamp is called an **archive Time Stamp**. Before its usage it is necessary to verify and supplement all signatures and Time Stamps that are being time stamped by the archive Time Stamp with CRL (OCSP) for their validity verification up to the certificate signed by the root certificate.

If the signature does not contain any other archive Time Stamp, then certificates validity of individual Time Stamps is verified in the interval beginning from time which the Time Stamp contains up to the current time and it is recommended to verify it with the most current data. It means, it is necessary to obtain the current CRL (OCSP) verified by the root certificate and then to look for CRL (OCSP) which is issued by the subordinate CA earlier but it must not be issued earlier than a time in the verified Time Stamp. It is necessary to continue by this process till CRL (OCSP) for verification of end user certificate is found. If the signature contains more archive Time Stamps then the interval for verification of archived Time Stamps validity is following (time from verified archived Time Stamp, time from the following archive Time Stamp).

The Archive Time Stamp must be created in the time when all certificates in the certification path are not expired and there is CRL (OCSP) for their verification. It is necessary because certificates can be verified retrospectively after archival Time Stamping.

A certification path is verified up to the certificate issued by the root CA in the verification process of data whose integrity is protected by the archive Time Stamp. The original root certificate could have already expired in time when the validity of the certification path is verified (at a long-time verification), thus it could have been deleted from trusted store being served for saving of the current trusted root certificate (trustAnchor – Distinguished Name + Public Key). Therefore it is necessary to obtain trusted information about which root certificate was defined as the trusted root certificate in verification time of archived data. Thus it is necessary to have recorded and published history of trusted root certificates.

An organization which performs CA accreditation and practices supervision should publish a history of trusted root certificates. Such organization has all necessary information needed for recording of historical data. Information about accredited Certification Service Providers and information about the history of accredited certification services providing in the form of TSL [12] are supposed to be used within the European Union. Until now there are not provided any clear rules for TSL issuance.

As the solution of this situation (within the Slovak Republic) it is possible to issue a list of already expired trusted root certificates in the form of archival signature from the individual root certificates. The rules defined in the NSA document “Trusted publication of the electronic documents lists” will be applicable for the publication of this list.

Signatures of accredited Certification Service Providers lists are verified to the current trusted root certificate. The records about accredited providers in the list contain even information about expired root certificates which were trusted in the past and which can be used for verification of the data which are protected by the archive Time Stamp.

## 9 Attributes of the certificate X.509 in the Certification Path Validation

Attributes from an issued certificate and an issuer certificate are checked for the conformity in the certification path building. If the issued certificate contains the attribute pair then it must be also contained in the issuer certificate, otherwise the certificate is invalid.

Each certification path building must meet at least the following conditions to be able to build the unambiguous certification path.

### 9.1 Sequence of verified items in the Certification Path Building

1. *Issuer* name in the verified certificate must be equal to Subject name in the issuer certificate.
2. Qualified certificate format must be equal to X.509 certificate v3 in DER coding.
3. The signature of the issued certificate must be verified by public key from the issuer certificate (in accordance with rules defined in the Slovak legislation, SHA-1 is recommended to be superseded by SHA-256 in the European Union).
4. In *AuthorityKeyIdentifier* of the verified certificate *keyIdentifier* must be equal to *SubjectKeyIdentifier* in the issuer certificate, each certificate must contain *SubjectKeyIdentifier*.
5. If the verified certificate contains *authorityCertIssuer* in *AuthorityKeyIdentifier* then the name must be equal to *Issuer* name in the issuer certificate.
6. If the verified certificate contains *authorityCertSerialNumber* in *AuthorityKeyIdentifier* then *authorityCertSerialNumber* must be equal to *serialNumber* in the issuer certificate.
7. Control time must be in the interval (*notBefore*, *notAfter*) – the certificate must not be expired in control time.
8. The certificate validity is verified on the basis of methods that are defined in Annex B.
9. There must an intersection exist of intervals of checked certificate validation (*notBefore*, *notAfter*) and the issuer certificate (*notBefore*, *notAfter*) where control time is in the mentioned interval of intersection.
10. Path length is checked in accordance with decreasing variable *maxPathLength*. If *BasicConstraints.pathLengthConstraint* is smaller than *maxPathLength* then *MaxPathLength* must be set on value *BasicConstraints.pathLengthConstraint* from the certificate. If *maxPathLength* is zero then CA certificate can not follow the certificate.
11. The issuer certificate must contain:
  - *BasicConstraints.Ca* setup on the value *TRUE*.
  - *KeyUsage* containing *keyCertSign*. If the certificate is used for the CRL signing then *KeyUsage* contains even *CRLSign*.
12. End entity qualified certificate must contain:
  - *KeyUsage* with *nonRepudiation* value and possibly even *digitalSignature*.
  - If the certificate is determined for Time Stamp verification then *extendedKeyUsage* must contain *id-kp-timeStamping*.
  - If the certificate is determined for OCSP verification then *extendedKeyUsage* must contain at least *id-kp-OCSPSigning*.
13. Qualified certificates, issued in accordance with the Slovak legislation, must contain in extension *CertificatePolicies* minimally OID *qualifiSK* (1 3 158 36061701 0 0 0 1 2 2). Qualified certificates of a natural person issued for SSCD contain OIDs *id-etsi-qcs-QcCompliance* and *id-etsi-qcs-QcSSCD* in extension *QCStatements* in accordance with ETSI document [6].

Qualified certificate of end entity (of a signer) issued by accredited CA must always contain at least the item *AuthorityKeyIdentifier.keyIdentifier*, in case of ambiguity of the certification path building all three items must be filled in.

Certificate extension id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::= { id-ce 35 }

```
AuthorityKeyIdentifier ::= SEQUENCE {
    keyIdentifier           [0] KeyIdentifier           OPTIONAL,
    authorityCertIssuer     [1] GeneralNames           OPTIONAL,
    authorityCertSerialNumber [2] CertificateSerialNumber OPTIONAL }
```

Following items are necessary to process and check:

- *Policy Mappings*
- *Name Constraints* - restricts the named subtree fields
- *Policy Constraints* - restricts the policy and mapping

Following items are helpful in signing and also in the certification path building and verification:

- *CRL Distribution Points* - contains paths for CRL obtaining
- *Authority Information Access* - contains paths for CA certificates and OCSP
- *Subject Information Access* - contains paths for TimeStamp and CA cert.

## 10 Certificate Path Validation

PVM procedure (PVM – Path Validation Module) is one of the application critical components that verify the certification path of X.509 certificates. The application can obtain the certification path for verification either from signature attributes or it builds the certification path on the basis of conditions for certification path building.

The result of PVM procedure for certification path validation must be identical with the result of the validation in accordance with the document RFC 3280 in section 6. The application does not have to use algorithms exactly according to the section 6 of the document RFC 3280 but results must be identical.

PVM will confirm to application the validity of an end entity certificate for chosen purpose from which certification paths start and end in certificates that are verified by the trusted root certificate, for example by CA root certificate of the NSA. In practice, it is possible to build more certification paths. Therefore the application SHOULD KNOW to process such situation and choose a suitable certification path.

The application can choose a suitable certification path on the basis of input conditions to PVM. PVM can have as an input the trusted root certificate and also the set of certification policies that indicates the suitable path. The inputs to the algorithm can be also obtained from the signature policy if it was used by the application. The choice of the path on the basis of the certification policy is determined by values in the certificate extensions, especially: *Certificate Policies*, *Policy Mappings*, *Policy Constraints* and *Inhibit Any-Policy*.

The set of acceptable certificate policies can enter PVM either:

1. Explicitly, e. g. from the signature policy item.
2. Or it can be required directly in the certificate in which the critical extension *PolicyConstraints requireExplicitPolicy SkipCerts* is set.

In the first case all certificates in the certification path are required to contain a non-empty intersection with the set of explicit certificate policies from signature policy entry.

In the second case certification policies from the certificate which contains the critical extension *PolicyConstraints requireExplicitPolicy* are required to be included in the certificates of the certification path directly or by means of *policyMappings* up to the verified certificate.

ETSI [6] identification of the qualified certificate that is found in the extension *qCStatements* identifies qualified certificates of a natural person issued for the public key to which a signer private key belongs. This signer private key is saved on the Secure Signature Creation Device SSCD:

```
id-etsi-qcs OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4) etsi(0)
id-qc-profile(1862) 1 }
```

```
id-etsi-qcs-QcSSCD OBJECT IDENTIFIER ::= { id-etsi-qcs 4 }
```

```
esi4-qcStatement-4 QC-STATEMENT ::= { IDENTIFIED BY id-etsi-qcs-QcSSCD }
```

An Identifier of the statement (represented by an OID), made by the CA, stating that the private key associated with the public key in the certificate is stored in a Secure Signature Creation Device according to Annex III of the EU Directive 1999/93/EC [1], as implemented in the law of the country where the CA is established.

## 10.1 The algorithm for Certification Path Building

The algorithm for certification path building uses input values from which some of them can be a part of the signature policy.

The list of data and rules required for certification path building:

- **End entity certificate** whose public key is used for the electronic signature verification.
- **UserInitialPolicy** a set of policies required by the user.
  - Its intersection with *certificatePolicies* is a non-empty set.
  - If *anyPolicy* is in *userInitialPolicy*, then all policies are acceptable.
- **ValidPolicySet** is a set of acceptable policies that is being formed as an intersection with the certificate items *certificatePolicies*. If the policy mapping is permitted then *validPolicySet* is being formed through *policyMapping*. At the beginning *validPolicySet* is not initialized and it will take items from the first certificate of the certification path which is verified by the root certificate.
- **Certificates list** that can be used for certification path building.
- **CRL (OCSP) list.**
- **Control time** in which the certification path is being built and checked (e. g. time of the oldest valid TimeStamp of the signature).
- **The list of implicitly trusted root CA certificates** in the form of *selfSigned* certificates which do not create the certification path but they contain trustAnchor (Distinguished Name and Public Key). The trusted trustAnchor does not have to be present only in the form of the certificate but it can also be found in TSL [12] as a separate pair (DN + Public Key).
- **PermittedSubtrees** is set of names for each name type defining a set of subtrees within which all subject names in subsequent certificates in direction from the root certificate in the certification path MUST fall. It is not initialized at the beginning and it accepts all names. Only items defined in *permittedSubtrees* are checked, thus the certificate can contain whatever else. The application must be able to process:
  - *subject - DistinguishedName*

- *subjectAltName.directoryName - DistinguishedName*
  - *subjectAltName.rfc822Name*
  - *subjectAltName.dNSName*
  - *subjectAltName.uniformResourceIdentifier*
  - *subjectAltName.iPAddress*
- ***ExcludedSubtrees*** is a set of names for each name type defining a set of subtrees within which no subject name in subsequent certificates in direction from the root certificate in the certification path may fall. The set is empty at the beginning of the algorithm. *PermittedSubtrees* verification does not influence *excludedSubtrees* verification. Only items defined in *excludedSubtrees* are checked, thus the certificate can contain whatever else. The application must be able to process:
    - *subject (DistinguishedName)*
    - *subjectAltName.directoryName (DistinguishedName)*
    - *subjectAltName.rfc822Name*
    - *subjectAltName.dNSName*
    - *subjectAltName.uniformResourceIdentifier*
    - *subjectAltName.iPAddress*
  - ***RequireExplicitPolicySkipCerts*** is a value that determines how many certificates will be skipped before the verification with *userInitialPolicy* and *validPolicySet* will be required. An initial value is set on path length +1. Following certificates can only decrease the value. If the value in *PolicyConstraints.requireExplicitPolicy* is smaller then *requireExplicitPolicySkipCerts* is set on this value. If the value of *requireExplicitPolicySkipCerts* is zero then for a current certificate and every following certificate in the certification path is required:
    - a non-empty set from the intersection of *userInitialPolicy* and the union (*policyIdentifier from CertificatePolicies*),
    - a non-empty intersection of *validPolicySet* and the union (*policyIdentifier from CertificatePolicies*).
  - ***InhibitPolicyMappingSkipCerts*** is a value that determines how many certificates will be skipped before the policy mapping will be forbidden. An initial value is set on path length +1. Following certificates can only decrease the value. If the value in *PolicyConstraints.inhibitPolicyMapping* is smaller then *inhibitPolicyMappingSkipCerts* is set on this value.
  - ***InhibitAnyPolicySkipCerts*** is a value that determines how many certificates will be skipped before *AnyPolicy* usage will be forbidden. An initial value is set on path length +1. Following certificates can only decrease the value. If the value in *InhibitAnyPolicy* is smaller then *InhibitAnyPolicySkipCerts* is set on this value. If the value and work of *AnyPolicy* is not mentioned in further parts of this document then *AnyPolicy* can be used with the union of the set of OIDs from *CertificatePolicies* extension while it is not forbidden by *InhibitAnyPolicySkipCerts*.
  - ***MaxPathLength*** is initialized to the value of the certification path length and it decreases in each CA certificate, except *selfSigned* certificate. Following certificates can only decrease the value. If the value in *BasicConstraints.pathLenConstraint* is smaller then *MaxPathLength* is set on this value. *MaxPathLength* determines how many more CA certificates can follow in this path.

## 10.2 Searching for certification paths through the recursive procedure

The algorithm for certification path searching must ensure a finding of a certification path up to the certificate which is verified by implicitly “trusted root certificate”. The algorithm must not stay in a dead lock, while searching for the certification path, in spite of the usage of a cross certification of CA certificates. If a local database does not contain an issuer’s certificate, it is possible to obtain the certificate from the address that is found in the certificate attribute *AuthorityInformationAccess*.

The certification path can be described by this process:

- for all certificates  $C_i$  where  $i$  is from  $\{1, \dots, n - 1\}$  is applied that a subject name of the certificate  $C_{i+1}$  is an issuer name of the certificate  $C_i$ ;
- the certificate  $C_n$ , is an implicitly trusted root certificate *selfSigned*, anyway this certificate is not taken into consideration in verification of *pathLenConstrain* – the length of the certification path and *NameConstraints* because it is not a part of the certification path;
- the certificate  $C_1$  is an end entity certificate that is verified and its public key is used for the verification, e.g. of the document signature or TimeStamp.

While the signature is verified, an end entity certificate is searched firstly on the basis of certificate identifiers: *issuer* name, *serialNumber*, *hash of certificate* and *SubjectKeyIdentifier*. The certificate suitability for required purpose is verified on the basis of certificate extensions, e. g. *KeyUsage*, *ExtendedKeyUsage*, *CertificatePolicies*, *QcStatements* or other extensions in accordance with the certificate usage type.

If the end entity certificate  $C_1$  with  $i=1$  was successfully obtained then it is followed by the recursive algorithm that tries to find a required certification path.

1. Try to find the certificate  $C_{i+1}$  to a certificate  $C_i$  which meets the conditions specified in the section 9.1 and which is not in the set of forbidden certificates or already contained in the certification path (except *selfSigned* certificate) or in previously found certification paths because we must prevent the algorithm from a recursive deadlock.
2. If the certificate  $C_{i+1}$  was not successfully found then save the certificate  $C_i$  in the set of forbidden certificates and decrease  $i$  by 1 and if  $i = 0$  then the algorithm is finished, otherwise find another certification path, e.g. through cross certificates beginning from the step no. 1.
3. If the certificate  $C_{i+1}$  is equal to the certificate  $C_i$  then the root certificate was found.
  - a. If the certificate  $C_i$  is implicitly trusted then save the certification path up to  $C_i$  in the list of found certification paths,  $i$  will be decreased by 1 and if  $i = 0$  then the algorithm is finished, otherwise find another certification path, e.g. through cross certificates beginning from the step no. 1.
  - b. If the certificate  $C_i$  is not implicitly trusted then save the certificate  $C_i$  in the set of forbidden certificates, decrease  $i$  by 1 and if  $i = 0$  then the algorithm is finished, otherwise find another certification path, e.g. through cross certificates beginning from the step no. 1.
4. Save the certificate  $C_{i+1}$  in the current certification path,  $i$  will be increased by 1 and continue with the step no. 1.

If the certification path is built or verified on the basis of the signature policy which contains the trusted root certificates list then implicitly trusted root certificates MUST be also found in the list of trusted root certificates that are saved in the signature policy. In other words, the implicitly trusted root certificate is a certificate to which the verifier trusts and at the same time it must be found in the signature policy.

If the list of found certification paths is not empty then the algorithm chooses the most suitable certification paths (e. g. on the basis of legal and technological requirements...) and verifies them by the algorithm defined in the following section 10.3., otherwise the procedure terminates with a failure indication about unsuccessful building of the certification path to the implicitly trusted root certificate.

### 10.3 Algorithm for Certificate Path Validation

An input into the process of validation can be through already prepared certification paths from electronic signatures or from the previous algorithm used for certification path searching. Each certification path will be verified separately.

The change of certificates order is a zero step in the certification path validation procedure:

- For all certificates  $C_i$  where  $i$  is from  $\{1, \dots, n-1\}$  is applied that a subject name of the certificate  $C_i$  is an issuer name of the certificate  $C_{i+1}$ .
- The certificate  $C_1$  is implicitly trusted root *selfSigned* certificate. This certificate does not belong to a certification path.
- The certificate  $C_n$  is an end entity certificate that is verified and its public key is used for the verification, e.g. of the document signature or TimeStamp.

Check the certification path with the certificates  $C_i$  where  $i = 1$  up to  $n$  and if any from the following conditions is not met then the algorithm terminates with the result INVALID, otherwise VALID.

1. If  $i = 1$  then
  - a. verify if the certificate  $C_i$  is *selfSigned* certificate and is implicitly trusted,
  - b. verify the certificate  $C_i$  with  $C_i$  in accordance with the method defined in the section 9.1. The points 4 to 9 are not realized in the verification process according to conditions from section 9.1 and only the variable *maxPathLength* is initialized in the point 10.
2. If  $i > 1$  then verify the certificate  $C_i$  with the certificate  $C_{i-1}$  in accordance with the method in the section 9.1.
3. If  $i = 1$  then this point does not have to be realized, otherwise search CRL or OCSP in the local database on the basis of the method defined in Annex B. If expected CRL or OCSP is not available, it is possible to obtain it from an address that is saved in *CRLDistributionPoints* or *AuthorityInformationAccess*. Verify the signature of CRL or OCSP with the certificate  $C_{i-1}$  (if CRL or OCSP are indirect then verification is realized by a different certification path) and verify the certificate validity  $C_i$  through CRL or OCSP.
4. If  $i$  is different from 1 then check the certificate  $C_i$  through *permittedSubtrees* in accordance with the section 10.1.
5. If  $i$  is different from 1 then check the certificate  $C_i$  through *excludedSubtrees* in accordance with the section 10.1.
6. If the certificate  $C_i$  contains *NameConstraints.permittedSubtrees* then save the intersection of *NameConstraints.permittedSubtrees* and *permittedSubtrees* in *permittedSubtrees*.
7. If the certificate  $C_i$  contains *NameConstraints.excludedSubtrees* then save the union of *NameConstraints.excludedSubtrees* and *excludedSubtrees* in *excludedSubtrees*.
8. The union of OIDs from the certificate  $C_i$  from items *CertificatePolicies* save in *explicitPolicies*.
9. If *requireExplicitPolicySkipCerts* = 0 then:

- a. a non-empty intersection must be found between *userInitialPolicy* and *explicitPolicies*,
  - b. a non-empty intersection must be found between *validPolicySet* and *explicitPolicies*.
10. Set *ValidPolicySet* on the value of the intersection of *validPolicySet* and *explicitPolicies*.
  11. If *inhibitPolicyMappingSkipCerts* > 0 then try to create a set of *mappedPolicies* by the process that for each pair from the certificate  $C_i$  *PolicyMappings(issuerDomainPolicy, subjectDomainPolicy)*, whose *issuerDomainPolicy* is found in the set of *validPolicySet*, add OID *subjectDomainPolicy* to *mappedPolicies* and in the end delete the policy *issuerDomainPolicy* from *validPolicySet*. Finally the union of *validPolicySet* and *mappedPolicies* save in *validPolicySet*.
  12. If *requireExplicitPolicySkipCerts* = 0 then a non-empty intersection must be found between *userInitialPolicy* and *validPolicySet*.
  13. If the certificate  $C_i$  contains *PolicyConstraints.requireExplicitPolicy* and the value is smaller than the value in *requireExplicitPolicySkipCerts*, set *requireExplicitPolicySkipCerts* on the value of *PolicyConstraints.requireExplicitPolicy*. Otherwise decrease *requireExplicitPolicySkipCerts* by one.
  14. If the certificate  $C_i$  contains *PolicyConstraints.inhibitPolicyMapping* and the value is smaller than the value in *inhibitPolicyMappingSkipCerts*, set *inhibitPolicyMappingSkipCerts* on the value of *PolicyConstraints.inhibitPolicyMapping*. Otherwise decrease *inhibitPolicyMappingSkipCerts* by one.
  15. If the certificate  $C_i$  contains *InhibitAnyPolicy* and the value is smaller than the value in *InhibitAnyPolicySkipCerts*, set *InhibitAnyPolicySkipCerts* on the value of *InhibitAnyPolicy*. Otherwise decrease *InhibitAnyPolicySkipCerts* by one.
  16. If *maxPathLength* is zero then only the end entity certificate  $C_{i=n}$  can follow.
  17. If the certificate  $C_i$  contains *BasicConstraints.pathLenConstraint* and *maxPathLength* is higher than *maxPathLength* is set on the value of *BasicConstraints.pathLenConstraint*. Otherwise *maxPathLength* is decreased by one.
  18. The certificate  $C_i$  must not contain unknown critical certificate extensions.
  19. Continue with processing of another certificate according to step 1.

## Annex A (informative) Verification on the basis of OCSP with positive response and with OCSP response in accordance with RFC 2560

### A.1 OCSP with positive response on the basis of data from database

Conditions for verification of the certificate validity by means of OCSP based on database are clear and simple because database contains necessary data:

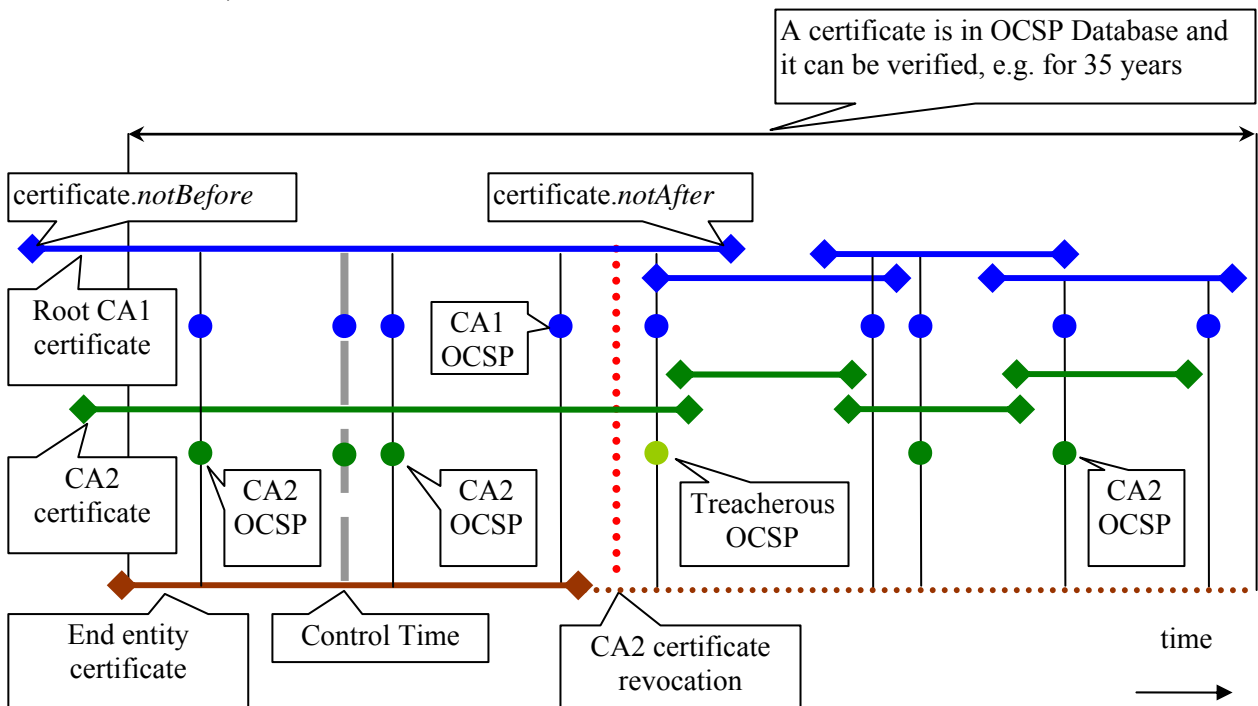
- a certificate;
- certificate status;
- time and reason of its revocation in case of a certificate revocation,

for a long period, e. g. for 35 years. The verifier can ask for a certificate status whenever without a necessity of the verification if the verified certificate had expired.

The positive answer about the certificate status is the main difference in comparison with OCSP (which takes information about the certificate invalidity from records in CRL) in accordance with RFC 2560. It is because the answer contains status of a certificate from which the hash value of the certificate is in the answer extension, thus the verifier can be sure that OCSP knows the certificate and the certificate status.

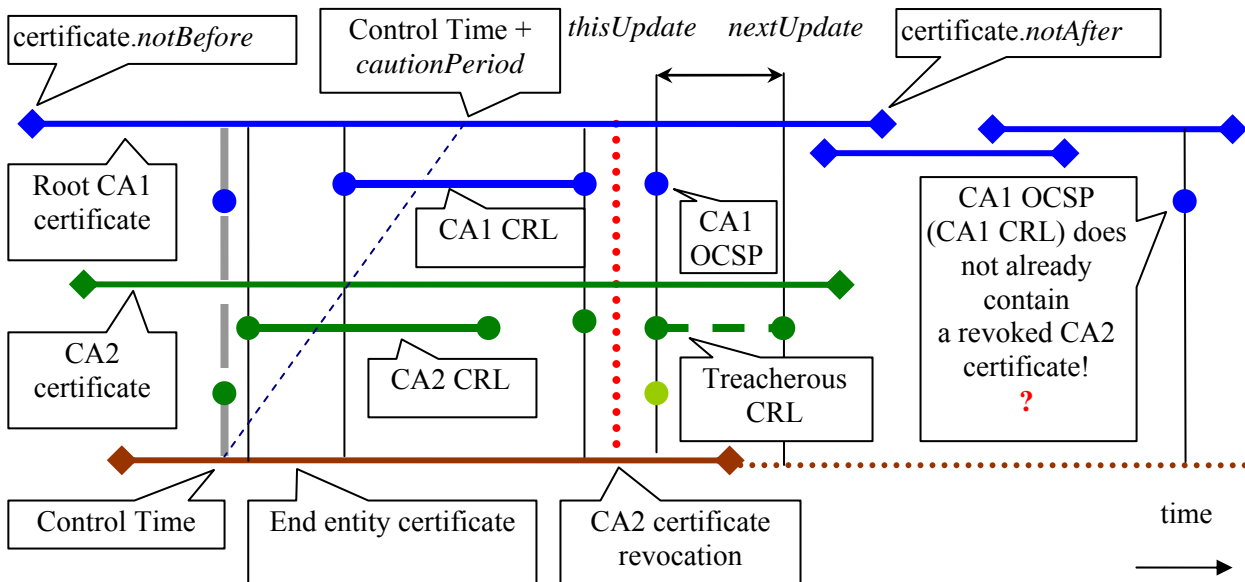
Another advantage is that it is possible to verify a status of expired certificate with OCSP whose signature is verified with the current certification path, where the first certificate in this certification path for OCSP signature validation is verified with the current valid trusted root certificate. Thus OCSP is verified to currently valid root certificate even if certificates status that OCSP returns are already expired for a long time together with the certification path.

OCSP, which is based on records in the database, must contain a *certHash* extension defined in ISIS-MTT Optional SigG-Profile. The *CertHash* extension ensures a positive statement that system which issues OCSP knows the certificate whose status the system returns (certificate was issued and is in the database).



## A.2 OCSP response in accordance with RFC 2560

If OCSP which achieves a status of the certificate validity from CRL is used then it is possible to use OCSP only in time in which data about a certificate status can be found in CRL. It has just one advantage: if CRL grows into the extreme size then OCSP answer will be smaller but on the other side constant requiring for the status through OCSP will increase demands for OCSP server capacity.



## Annex B (normative) Verification of a certificate validity

### B.1 Verification on the basis of OCSP

Certificate validity conditions for verification by means of OCSP. The OCSP response must contain a status which was known after the time *certificate.notBefore* of a verified certificate or in time when information about the certificate status is available for OCSP even after the expiration of the certificate. Thus OCSP contains *ArchiveCutoff* with the value which is smaller than the certificate expiration time or a positive statement in the form of *CertHash* in the extension of the OCSP response about the situation that OCSP knows the certificate and its status.

**Table 1 Verification with OCSP**

1. **if** ( *certificate.notBefore* < *OCSP[certificate].thisUpdate* ) **and**  
 ( ( *OCSP.ArchiveCutoff* <= *certificate.notAfter* ) **and** ( 0 < *OCSP.ArchiveCutoff* ) **or**  
 ( *OCSP[certificate].thisUpdate* <= *certificate.notAfter* ) **and** ( 0 = *OCSP.ArchiveCutoff* ) **or**  
 ( *OCSP[certificate].CertHash* = *certificate.CertHash* ) ) **then**
2.     **if** *OCSP[certificate].CertStatus* = *good* **then**
3.         **If** ( *ControlTime* + *cautionPeriod* ) <= *OCSP[certificate].thisUpdate* **then**  
            **VALID**
4.         **else**  
            **INCOMPLETE VERIFICATION – obtaining of a new OCSP response**
5.     **else**  
    **if** *OCSP[certificate].CertStatus* = *revoked* **then**  
        **if** *Control Time* < *OCSP[certificate].revocationTime* **then**  
            **VALID**
6.         **else**  
            **INVALID**
7.     **else**  
        **INCOMPLETE AUTOMATIC VERIFICATION – OCSP does not know the current certificate status because *OCSP[certificate].CertStatus* = *unknown***  
        **It is necessary to obtain OCSP from another address or to verify on the basis of CRL.**
8. **else**  
    **INCOMPLETE AUTOMATIC VERIFICATION – request to CA for CRL or for OCSP issued in period of a certificate validity + a period of time in which the record about the certificate revocation in CRL or OCSP is still present.**

Where:

- *OCSP.ArchiveCutoff* - if *ArchiveCutoff* is not in the OCSP response then its value is 0, otherwise the value saved in *ArchiveCutoff* is defined in accordance with RFC 2560.
- *OCSP[certificate].CertHash* is a certificate hash whose status OCSP returns (ISIS-MTT private extensions). If this extension is found in OCSP then the extension creates positive information about the fact that OCSP knows the certificate and the status of the verified certificate.
- *Certificate.CertHash* is a certificate hash whose validity is verified.
- *OCSP.producedAt* is the time of the OCSP issuing.
- *OCSP[certificate].thisUpdate* is the time by which the correct information about the certificate status was known.

- Certificate.*notBefore* is the time of the beginning of verified certificate validity. Certificate.*notAfter* is the time after which the certificate will be expired.
- OCSP[certificate].*revocationTime* is a date of the certificate revocation.
- OCSP[certificate].*CertStatus* is a certificate status in OCSP which can have only 3 values.

Explanations to the following conditions:

1. OCSP is issued in time of certificate validity + a period of time during which the record about the certificate revocation for OCSP is known even after the certificate expiration.
2. The certificate was not revoked; it is not in OCSP.
3. A certificate status in OCSP is known after control time. The *CautionPeriod* can be set on the value “zero” (the result of the verification will be the same as if still waiting for *cautionPeriod*) during the verification of the certificate issued in accordance with the rules of the Slovak legislation (OID of the certification policy 1.3.158.36061701.0.0.0.1.2.2).
4. The certificate status in OCSP is not known after control time. It is necessary to ask for a new OCSP.
5. The certificate was revoked after control time, thus it is valid.
6. The certificate is revoked in OCSP before control time.
7. OCSP is not able to determine a certificate status, it is necessary to try other OCSP or CRL.
8. It is necessary to obtain OCSP or CRL issued in time when the certificate has not been expired yet + a period of time in which the certificate status is still known in OCSP or CRL. It is issued after control time.

## B.2 Verification on the basis of CRL

Certificate validity conditions for verification by means of CRL. The CRL answer must contain a status which was known after the time certificate.*notBefore* of the verified certificate or in time when information about the certificate status is available in CRL even after the certificate expiration. Thus CRL contains *expiredCertsOnCRL* with the value that is smaller than the certificate expiration time.

**Table 2 Verification with CRL**

1. **if** ( certificate.*notBefore* < CRL.*thisUpdate* ) **and**  
 ( (CRL.*expiredCertsOnCRL* <= certificate.*notAfter* ) **and** ( 0 < CRL.*expiredCertsOnCRL* ) **or**  
 ( CRL.*thisUpdate* <= certificate.*notAfter* ) **and** ( 0 = CRL.*expiredCertsOnCRL* ) ) **then**
2.     **if** certificate **is not in** CRL **then**
3.         **If** (ControlTime + *cautionPeriod* ) <= CRL.*thisUpdate* **then**  
            **VALID**
4.         **else**  
            **INCOMPLETE VERIFICATION – waiting for a new CRL**
5.     **else**  
        **if** ControlTime < CRL[certificate].*revocationDate* **then**  
            **VALID**
6.         **else**  
            **INVALID**
7. **else**  
        **INCOMPLETE AUTOMATIC VERIFICATION – request to CA for CRL issued in time of a certificate validity + a period of time in which the entry about the certificate revocation in CRL is still present.**

Where:

- If *CRL.expiredCertsOnCRL* is not present in the CRL extension then its value is 0, otherwise the value is according to ITU-T Rec. X.509 (08/2005) [10].
- *CRL.thisUpdate* is the time by which the correct information about the certificate status was known.
- Certificate.*notBefore* is the time of the beginning of verified certificate validity.
- Certificate.*notAfter* is the time after which the certificate is expired.
- *CRL[certificate].revocationDate* is a date of the certificate revocation in CRL.

Explanations to the following conditions:

1. CRL is issued in time of certificate validity + a period of time during which the record about the certificate revocation is known in CRL even after the certificate expiration.
2. The certificate was not revoked; it is not in CRL.
3. The certificate status in CRL is known after control time. The *CautionPeriod* can be set on the value “zero” (the result of the verification will be the same as if still waiting for *cautionPeriod*) during the verification of the certificate issued in accordance with the rules of the Slovak legislation (OID of the certification policy 1.3.158.36061701.0.0.0.1.2.2).
4. CRL is not issued after control time, and it is necessary to wait for a new CRL.
5. The certificate was revoked after control time, thus it is valid.
6. The certificate is revoked before control time in CRL.
7. It is necessary to obtain CRL issued in time when the certificate has not been expired yet + a period of time in which the certificate status is still known in CRL. It is issued after control time.

## **Annex C (informative) Revisions made since previous version**

### **C.1 Additional requirements**

The following items have been added which significantly affect the requirements:

The verification on the basis of `CRL.expiredCertsOnCRL` was added to the B.2.

The conclusion of the section 7 was removed and added to the section 8. The section 8 defines methods in the archiving.

### **C.2 Updated requirements**

The following items have been updated to extend choices or otherwise modify requirements:

The section 3 was removed to the section 5. The root certificate is not a part of the certification path. It is defined unambiguously in the section 5.

The requirements defined in the section 5 were united. It is described in the Annex B.

### **C.3 Clarifications**

The following items have been updated to clarify existing requirements:

*Control Time* is more specified.

The relationship between *grace period* and *caution period* is defined unambiguously.

More detailed explanations were added to sections 9 and 10.

### **C.4 Editorial**

A number of other editorial changes were made which do not affect the technical content of the present document:

The preface in the section 5 was reduced and removed to the Annex A.

The section 6 was deleted.

The numeration has increased by one from the section 8.

## Annex D (informative) The list of used literature

Basic documents for the Electronic Signature in accordance with the legislation of the Slovak Republic

<http://www.nbusr.sk/en/electronic-signature/legislation/index.html>

Qualified electronic signature formats

<http://www.nbusr.sk/en/electronic-signature/approved-formats/index.html>

Certification path building and validity verification of certificates

<http://www.nbusr.sk/en/electronic-signature/verification/index.html>

ETSI TS 102 176-1: "Electronic Signatures and Infrastructures (ESI); Algorithms and Parameters for Secure Electronic Signatures; Part 1: Hash functions and asymmetric algorithms".

ETSI TS 101 903 "XML Advanced Electronic Signatures (XAdES)."

ETSI TR 102 041 "Signature Policies Report"

ETSI TS 102 231 V2.1.1 (2006-03) Electronic Signatures and Infrastructures (ESI); Provision of harmonized Trust-service status information

US Secure Hash Algorithms (SHA and HMAC-SHA)

<http://www.rfc-archive.org/getrfc.php?rfc=4634>

Internet X.509 Public Key Infrastructure: Certification Path Building

<http://www.rfc-archive.org/getrfc.php?rfc=4158>

NIST X.509 path validation test suite

<http://csrc.nist.gov/pki/testing/x509paths.html>

<http://csrc.nist.gov/pki/testing/pathdiscovery.html>

## Annex E History

<b>Version:</b>	<b>Date of issuing:</b>	<b>Note:</b>	<b>Editor:</b>
Version 1.1	29.9.2005	First issuance, revoked	Ing. Peter Rybár, NSA
Version 1.2	16.10.2005	Adding of the OCSP	Ing. Peter Rybár, NSA
Version 1.2.1	24.11.2005	Simplifying of the conditions	Ing. Peter Rybár, NSA
Version 1.3 Č.: 1891/2006/IBEP-001	19.3.2006	Specifying OCSP more precisely, TSP verification and policy mappings	Ing. Peter Rybár, NSA
Version 1.4 Č.: 1891/2006/IBEP-011	19.11.2006	Completing and making clear of the conditions for ensuring of an interoperability	Ing. Peter Rybár, NSA