COMMON PKI SPECIFICATIONS
FOR INTEROPERABLE APPLICATIONS

FROM T7

CORRIGENDA

TO

COMMON PKI SPECIFICATION 2.0

AS OF 20 JANUARY 2009

VERSION 1.0 – 30 NOVEMBER 2010

Contact Information

The up-to-date version of the Common PKI specification can be downloaded from www.common-pki.org or from www.common-pki.de

Please send comments and questions to common-pki@common-pki.org.

Document History

VERSION DATE	CHANGES
1.0 30 November 2010	Initial Corrigenda to Common PKI Specification v2.0 The following issues are addressed: • Profile the *ESSCertIDv2* field within the *signingCertificateV2* CMS Attribute as introduced by RFC 5035 in accordance with the established Common PKI profile of the predecessor *signingCertificate* and *ESSCertID*. • Complete the update from *signingCertificate* / *ESSCertID to signingCertificateV2* / *ESSCertIDv2* in requirements on file signatures. • Mark the possibility of providing an explicit reference point in time as input to the certificate validation algorithm as Common PKI profiling, as the RFC 5280 validation algorithm always checks for its time of execution. • Correct the W3C URI for SHA-384. • Reflect the Common PKI 2.0 hash algorithm requirements according to Part 6 in the corresponding requirements of Part 8. Correct Algorithm URIs and schema references.

Table of Contents

1 Preface

This document contains a list of corrigenda to correct and clarify the Common PKI Specification v2.0.

The corrigenda become immediately effective with the publication of this document, i.e. the effectual text of the Common PKI specification will be that of the Common PKI Specification v2.0 as of January 20th, 2009 *with the changes specified in this document applied*.

Changes and additions are highlighted by background colour, deletions by ~~background colour and crossed out~~.

2 Corrigenda to Part 1: Certificate and CRL Profiles

Currently no corrigenda.

3 Corrigenda to Part 2: PKI Management

Currently no corrigenda.

4 Corrigenda to Part 3: CMS based Message Formats

1) In P3.T5#9 add

1	*signingCertificateV2* *id-aa-signingCertificateV2* {1 2 840 113549 1 9 16 2.47}	Sequence of certificate identifiers starting with the certificate of the signer	[RFC 5035]	3	+-		+-	+-	The *issuerSerial* field of the *ESSCertIDv2* within *signingCertificateV2* MUST not be empty.	[5]

2) Change P3.4.1 to

[~~RFC~~ RFC3852] allows including attribute certificates in the certificate list. For all attribute certificates, which are intended by the signer to be used for the signature, a reference MUST be included in the *signedAttributes* of the corresponding *SignerInfo* using the *SigningCertificateV2* attribute. The *issuerSerial* field of the *ESSCertIDv2* within *SigningCertificateV2* MUST not be empty. These informations are intended for the recipient, so that all certificates required for the verification of the file signature can easily be obtained. Note that certificates provided in the 'certificates' field are not part of the signed content and are thus not protected against substitution attacks.

5 Corrigenda to Part 4: Operational Protocols

Currently no corrigenda.

6 Corrigenda to Part 5: Certificate Path Validation

3) Change P5.T2#1 to

1	`bool ValidateCertificate(`
	`  CertInfo  in  tbvCert,`
	`  CertInfoList  in  tbvCerts,`
	`  KeyPurpose  in  intendedKeyUsage,`
	`  Time  in  refTime,`
	`  PolicyConstraints  in  initialPolicySet,`
	`  CertInfoList  inout  trustedCerts,`
	`  CrlInfoList  inout  trustedCrls )`
	`{`
		This is the main entry point of the certificate path validation algorithm.
		The 'to be verified' target certificate or attribute certificate is passed in *tbvCert*.
		tbvCerts may contain zero or more certificates – other than the 'to be verified' certificate – of a path to some root certificate. Most commonly, *tbvCerts* contains certificates trusted by the signing/decrypting party, but not necessarily trusted by the relying party.
		The required usage of the certified key is indicated in *intendedKeyUsage*. In case of an attribute certificate, this parameter is ignored by the procedure.
		~~The point in time, to which status information should be obtained, is passed in *refTime*. It may be the current time (typical for mail authentication, encryption) or some point in the past (typical for non-repudiation service).~~
		pathConstraints conveys input parameters from the relying application to the basic path validation algorithm (BPVA). These parameters contain policy constraints or naming constraints that have to be verified during path validation.
		trustedCerts MUST contain at least one trusted self-signed root certificate and may contain further CA and EE certificates, all of which having a path to one of those trusted root certificates. These certificates are typically stored on the local system to accelerate the validation procedure. *trustedCerts* may further contain cross-certificates (issued by a trusted CA to some other CA), each having a valid path to one of those root certificates.
		trustedCrls may contain complete CRLs that have previously been downloaded, successfully verified and stored in the local database. This storage allows a reuse of complete CRLs in later validations without needing to access the directory service. *trustedCrls* may furthermore contain complete CRLs that are locally maintained, e.g. by regularly downloading delta-CRLs from an LDAP-Server or by obtaining the list by some out-of-band mechanism (e.g. unsigned CRLs of root certificates).
		This function returns *true* if the certificate has been successfully verified, including mathematical verification, constraint and status checking; respectively *false* if mathematical check failed, some constraint is not met, a relevant certificate cannot be obtained or has been revoked, status information cannot been obtained or no certification path could have been built to any of the trusted root certificates.
		trustedCerts will be updated with the certificates of a successfully validated path to		

		allow local storage and reuse of validated certificates and corresponding status information. *trustedCrls* will be similarly updated with verified CRLs. **Common PKI Profile:** The point in time, to which status information should be obtained, is passed in *refTime*. It may e. g. be the current time (typical for mail authentication, encryption) or some point in the past (typical for non-repudiation service).		

7 Corrigenda to Part 6: Cryptographic Algorithms

4) In P6.T1#2a replace

2a	SHA -384	one-way hash function	[RFC 4055] [XML_ENC] [FIPS 180-2]		n. a.	+	+	OID: 2.16.840.1.101.3.4.2.2 http://www.w3.org/2001/04/~~xmlenc~~#sha384	[4]

by

2a	SHA -384	one-way hash function	[RFC 4055] [XML_ENC] [FIPS 180-2]		n. a.	+	+	OID: 2.16.840.1.101.3.4.2.2 http://www.w3.org/2001/04/xmldsig-more#sha384	[4]

8 Corrigenda to Part 7: Signature API

Currently no corrigenda.

9 Corrigenda to Part 8: XML based Message Formats

1) Change P8.T3#4 to

4	<attribute name="Algorithm" type="anyURI" use="required"/>	<xsd:enumeration value="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />	++	++	4.3.2	[2]
		<xsd:enumeration value="http://www.w3.org/2001/04/~~xmlenc#ripemd160~~xmldsig-more/rsa-ripemd160" />	-	~~-~~+		[2] [3]
		<xsd:enumeration value="http://www.w3.org/2000/09/xmldsig#dsa-sha1" />	++	++		[2]
		<xsd:enumeration value="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" />	+-	+		[2]
		<xsd:enumeration value="http://www.w3.org/2001/04/xmldsig-more#rsa-sha384" />	+-	+		[2]
		<xsd:enumeration value="http://www.w3.org/2001/04/xmldsig-more#rsa-sha512" />	+-	+		[2]

2) Change P8.T3.[2] to

[2]	Delimits the possible algorithms toDSA-SHA1, RSA-SHA1, ~~and~~ RSA-RIPEMD160, RSA-SHA256, RSA-SHA384 and RSA-SHA512.

3) Change P8.T5#3 to

3 ~~4~~	<attribute name="Algorithm" type="anyURI" use= "required"/>	<xsd:enumeration value="http://www.w3.org/2000/09/xmldsig#sha1" />	++	++	4.3.3.5	[1]
		<xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#ripemd160" />	-	~~-~~+		[1] [2]
		<xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#sha256" />	+	+		[1]
		<xsd:enumeration value="http://www.w3.org/2001/04/xmldsig-more#sha384" />	+	+		[1]
		<xsd:enumeration value="http://www.w3.org/2001/04/xmlenc#sha512" />	+	+		[1]

4) Change P8.T5.[1] to

[1]	Delimits the possible algorithms to SHA-1, ~~and~~ RIPEMD160, SHA-256, SHA-384 and SHA-512.

5) In P8.5.1 add

```
<xsd:restriction base="xsd:anyURI">
        <xsd:enumeration
                value="http://www.w3.org/2000/09/xmldsig#sha1" />
        <xsd:enumeration
                value="http://www.w3.org/2001/04/xmlenc#sha256" />
        <xsd:enumeration
                value="http://www.w3.org/2001/04/xmldsig-more#sha384" />
        <xsd:enumeration
                value="http://www.w3.org/2001/04/xmlenc#sha512" />
        <xsd:enumeration
                value="http://www.w3.org/2001/04/xmlenc#ripemd160" />
</xsd:restriction>
```

6) In P8.5.1 change

```
<xsd:import namespace="http://www.w3.org/2001/04/xmlenc#"
        schemaLocation="~~oscienc.xsd~~" />
```

to

```
<xsd:import namespace="http://www.w3.org/2001/04/xmlenc#"
        schemaLocation="xenc-schema.xsd" />
```

10 Corrigenda to Part 9: SigG-Profile

Currently no corrigenda.