COMMON ISIS-MTT SPECIFICATIONS
FOR INTEROPERABLE PKI APPLICATIONS

FROM T7 & TELETRUST

CORRIGENDA

TO

ISIS-MTT SPECIFICATION 1.1

AS OF 16 MARCH 2004

VERSION 1.2 – 18 JANUARY 2008

Contact Information

The up-to-date version of ISIS-MTT can be downloaded from www.isis-mtt.org

Please send comments and questions to experts@isis-mtt.org

Document History

VERSION DATE	CHANGES
1.0 22. July 2004	Initial Corrigenda to ISIS-MTT Specification v1.1 The following issues are addressed: • Processing support for indirect CRLs is not REQUIRED, but RECOMMENDED in the core part of ISIS-MTT (board decision of 27. May 2004). • Clarification: In accordance with RFC 3280, the *indirectCRL* flag must be set in the *IssuingDistributionPoint* extension of all indirect CRLs . • Several editorial changes and corrections.
1.1 15 Sept. 2006	Added a provision for the case of CRL-issuer names changing over time (thanks to Georgios Raptis).
1.2 18 Jan. 2008	Added provisions for use of SHA-256 and SHA-512 in ASN.1 data elements. Clarified wording of indirect CRL related corrigenda. Reflected *nonRepudiation* / *contentCommitment* renaming by ITU-T. Corrected an error in the explanation about *CRLDistributionPoints*. Relaxed requirement for registration of naming authorities for *admission* attributes.

Table of Contents

1 Preface

This document contains a list of corrigenda to correct and clarify the ISIS-MTT Specification v1.1.

The corrigenda become immediately effective with the publication of this document, i.e. the effectual text of the ISIS-MTT specification will be that of the ISIS-MTT Specification v1.1 as of March 16th, 2004 *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

1) In P1.T22.[2] add

[2]	Notes on support: [RFC3280]: it is RECOMMENDED always to include this extension in certificates. If no *cRLIssuer* is specified, the CRL MUST be issued by the issuer of the revoked certificates in the CRL. (Otherwise we speak about an *indirect* CRL.) If the certificate issuer is also the CRL issuer, then the *cRLIssuer* field MUST be omitted and the *distributionPoint* field MUST be present. **ISIS-MTT PROFILE:** Compliant CAs MUST issue CRLs and publish them via an LDAP-server. In addition to the LDAP service, the CA MAY publish CRLs via HTTP for cases, where some targeted clients cannot access the LDAP service (e.g. because of a local firewall policy). The CDP extension MAY contain more than one CDP. These have to be interpreted as alternatives. If access to a specific CDP fails, clients MAY try to access other alternatives. Delta-CRLs, if present in a CDP, MUST be present at the same location as the complete CRL. In the case of segmented CRLs, all segments MUST be present at the CDP. Basically, there are two different types of CRLs: 1) "*direct*" CRL: the CA that issued the certificate issues the corresponding CRLs too. In this case, if the *CRLDistributionPoints* is not included, the CRL MUST be located at the same LDAP node (in the *certificateRevocationLists* attribute) as the CA certificate. If it is located at another LDAP node or in another attribute, the corresponding DName (relative to the CA-node or absolute in the same directory) or LDAP-URL MUST be supplied in the *distributionPoint* field. Following [RFC3280], the *cRLIssuer* field MUST NOT be present in this "direct" case. 2) *indirect* CRLs are issued, i.e. the CRLs are signed with a key different from the key of the CA. In this case, the *CRLDistributionPoints* extension MUST be present and MUST include the *cRLIssuer* field containing the *subject* DName of the CRL-issuer and resp. of its signing certificate. The *distributionPoint* field MAY be present, pointing to the CRL (via a DName relative to the node of the CRL-issuer or absolute in the same directory; or via an URL). If the *distributionPoint* field is absent, the CRL MUST be located at the node of the CRL-issuer (in the *certificateRevocationLists* attribute). For the sake of vertical interoperability, it is RECOMMENDED that conforming applications process indirect CRLs in order to validate the revocation status of certificates. Indirect CRLs are frequently encountered in the domain of qualified certificates, where, however, the preferred mechanism of revocation checking is OCSP instead of CRL checking. Therefore support for indirect CRLs is not REQUIRED for applications adhering to the ISIS-MTT core standard (see the ISIS-MTT SigG profile for requirements on SigG-conforming applications).

2) In P1.T33#5 replace

5	IssuingDistributionPoint	{2 5 29 28}	Indicates whether the CRL covers revocations for end entity certificates only, for CA certificates only or for a limited set of reason codes and whether it is an indirect CRL.	++	+-	+	5.2.5	T36	

with

5	IssuingDistributionPoint	{2 5 29 28}	Indicates whether the CRL covers revocations for end entity certificates only, for CA certificates only or for a limited set of reason codes and whether it is an indirect CRL.	++	+-/++	+	5.2.5	T36	

3) Change P1.T33.[2] to

[2]	**ISIS-MTT PROFILE:** As readily described in T22.[2], there are two types of CRLs: 1) "*direct*" CRL: the CA that issued the certificate issues the corresponding CRLs too. This situation can be recognized by relying software if the following conditions apply: a. if the *CRLDistributionPoints* extension is missing from the ~~CA~~ certificate or b. it is present, but the *cRLIssuer* field is missing. 2) *indirect* CRL: the CRLs are signed with a key different from the key of the CA. This situation can be recognized by relying software if the *CRLDistributionPoints* extension is present in the ~~CA~~ certificate and the *cRLIssuer* field holds a DName (different from the *subject* of the CA certificate). Additionally, indirect CRLs MUST include an *IssuingDistributionPoint* extension with *indirectCRL* flag set to true. So that relying software can locate the certificate of the issuer of an indirect CRL, *AuthorityKeyIdentifier* MUST and *IssuerAltNames* MAY be included in indirect CRLs. The *IssuerAltNames* extension MAY contain the LDAP-URL of the node that holds the CRL-signer's certificate.

4) In P1.T12 add

3	nonRepudiation (1),	signature verification corresponding to non-repudiation service	+-	++			[3]

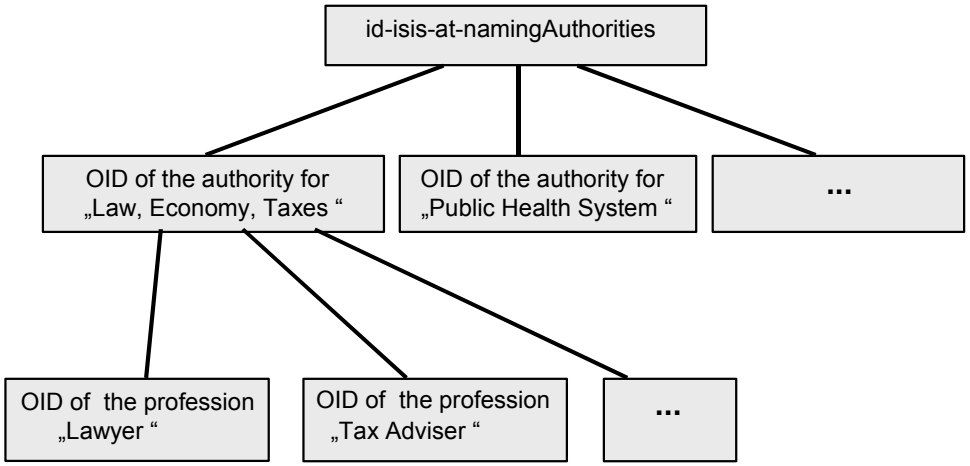
[3]	In April 2004 the ITU-T working group on X.509 renamed – without affecting its semantics – bit 1 of the *KeyUsage* extension to *contentCommitment* and declared the previous identifier *nonRepudiation* as being deprecated. **ISIS-MTT PROFILE:** Both identifiers will be treated as synonyms.

5) In P1.T29b replace

[1]	**ISIS-MTT PROFILE:** The relatively complex structure of *AdmissionSyntax* supports the following concepts and requirements:

- External institutions (e.g. professional associations, chambers, unions, administrative bodies, companies, etc.), which are responsible for granting and verifying professional admissions, are indicated by means of the data field *admissionAuthority*. An admission authority is indicated by a *GeneralName* object. Here an X.501 directory name (*distinguished name*) can be indicated in the field *directoryName*, a URL address can be indicated in the field *uniformResourceIdentifier*, and an object identifier can be indicated in the field *registeredId*.
- The names of authorities which are responsible for the administration of title registers are indicated in the data field *namingAuthority*. The name of the authority can be identified by an object identifier in the field *namingAuthorityId*, by means of a text string in the field *namingAuthorityText*, by means of a URL address in the field *namingAuthorityUrl*, or by a combination of them. For example, the text string can contain the name of the authority, the country and the name of the title register. The URL-option refers to a web page which contains lists with „officially" registered professions (text and possibly OID) as well as further information on these professions. Object identifiers for the component *namingAuthorityId* ~~are~~ grouped under the OID-branch *id-isis-at-namingAuthorities* and ~~must~~ be applied for.
  See ~~http://www.teletrust.de/anwend.asp?Id=30200&Sprache=E_&HomePG=0~~ for an application form and ~~http://www.teletrust.de/links.asp?id=30220,11~~ for an overview of registered naming authorities.
- By means of the data type *ProfessionInfo* certain professions, specializations, disciplines, fields of activity, etc. are identified. A profession is represented by one or more text strings, resp. profession OIDs in the fields *professionItems* and *professionOIDs* and by a registration number in the field *registrationNumber*. An indication in text form ~~must~~ always be present, whereas the other indications are optional. The component *addProfessionInfo* ~~may~~ contain additional application-specific information in DER-encoded form.

By means of different *namingAuthority*-OIDs or profession OIDs hierarchies of professions, specializations, disciplines, fields of activity, etc. can be expressed as illustrated in the figure below. The issuing admission authority ~~should~~ always be indicated (field *admissionAuthority*), whenever a registration number is presented. Still, information on admissions ~~can~~ be given without indicating an admission or a naming authority by the exclusive use of the component *professionItems*. In this case the certification authority is responsible for the verification of the admission information.

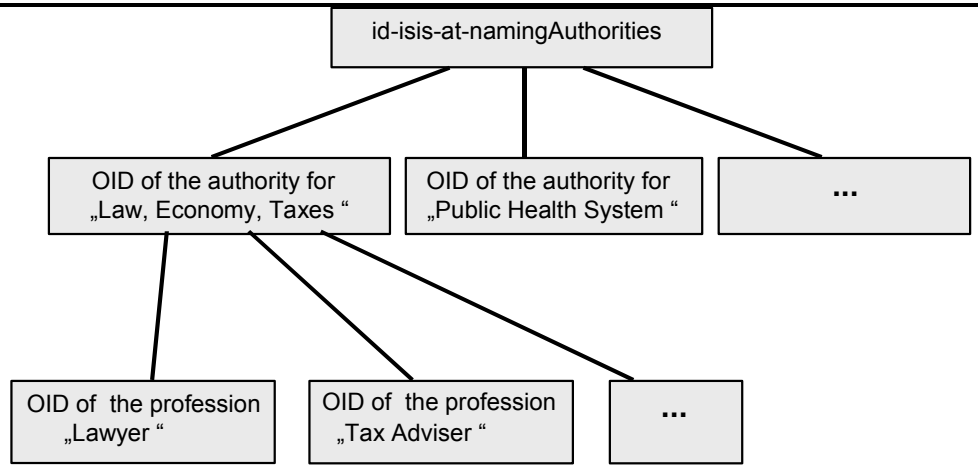


This attribute is *single-valued*. Still, several admissions can be captured in the sequence structure of the component *contentsOfAdmissions* of *AdmissionSyntax* or in

	the component *professionInfos* of *Admissions*. The component *admissionAuthority* of *AdmissionSyntax* serves as default value for the component *admissionAuthority* of *Admissions*. Within the latter component the default value can be overwritten, in case that another authority is responsible. The component *namingAuthority* of *Admissions* serves as a default value for the component *namingAuthority* of *ProfessionInfo*. Within the latter component the default value can be overwritten, in case that another naming authority needs to be recorded. The length of the string objects is limited to 128 characters. It is ~~recommended~~ to indicate a *namingAuthorityURL* in all issued attribute certificates. If a *namingAuthorityURL* is indicated, the field *professionItems* of *ProfessionInfo* ~~should~~ contain only registered titles. If the field *professionOIDs* exists, it has to contain the OIDs of the professions listed in *professionItems* in the same order. In general, the field *professionInfos* ~~should~~ contain only one entry, unless the admissions that are to be listed are logically connected (e.g. they have been issued under the same admission number).

by

[1]	**ISIS-MTT PROFILE:** The relatively complex structure of *AdmissionSyntax* supports the following concepts and requirements: • External institutions (e.g. professional associations, chambers, unions, administrative bodies, companies, etc.), which are responsible for granting and verifying professional admissions, are indicated by means of the data field *admissionAuthority*. An admission authority is indicated by a *GeneralName* object. Here an X.501 directory name (*distinguished name*) can be indicated in the field *directoryName*, a URL address can be indicated in the field *uniformResourceIdentifier*, and an object identifier can be indicated in the field *registeredId*. • The names of authorities which are responsible for the administration of title registers are indicated in the data field *namingAuthority*. The name of the authority can be identified by an object identifier in the field *namingAuthorityId*, by means of a text string in the field *namingAuthorityText*, by means of a URL address in the field *namingAuthorityUrl*, or by a combination of them. For example, the text string can contain the name of the authority, the country and the name of the title register. The URL-option refers to a web page which contains lists with „officially“ registered professions (text and possibly OID) as well as further information on these professions. Object identifiers for the component *namingAuthorityId* MAY be grouped under the OID-branch *id-isis-at-namingAuthorities* and MAY be applied for by interested authorities. See http://www.teletrust.de/fileadmin/files/oid/oid_Antrag.pdf for an application form and http://www.teletrust.de/index.php?id=524 for an overview of registered naming authorities. • By means of the data type *ProfessionInfo* certain professions, specializations, disciplines, fields of activity, etc. are identified. A profession is represented by one or more text strings, resp. profession OIDs in the fields *professionItems* and *professionOIDs* and by a registration number in the field *registrationNumber*. An indication in text form MUST always be present, whereas the other indications are optional. The component *addProfessionInfo* MAY contain additional application-specific information in DER-encoded form. By means of different *namingAuthority*-OIDs or profession OIDs hierarchies of professions, specializations, disciplines, fields of activity, etc. can be expressed as illustrated as a possible example in the figure below. The issuing admission authority SHOULD always be indicated (field *admissionAuthority*), whenever a registration number is presented. Still, information on admissions MAY be given without indicating an admission or a naming authority by the exclusive use of the component *professionItems*. In this case the certification authority is responsible for the verification of the admission information.



This attribute is *single-valued*. Still, several admissions can be captured in the sequence structure of the component *contentsOfAdmissions* of *AdmissionSyntax* or in the component *professionInfos* of *Admissions*.
The component *admissionAuthority* of *AdmissionSyntax* serves as default value for the component *admissionAuthority* of *Admissions*. Within the latter component the default value can be overwritten, in case that another authority is responsible.
The component *namingAuthority* of *Admissions* serves as a default value for the component *namingAuthority* of *ProfessionInfo*. Within the latter component the default value can be overwritten, in case that another naming authority needs to be recorded.

The length of the string objects is limited to 128 characters. It is RECOMMENDED to indicate a *namingAuthorityURL* in all issued attribute certificates. If a *namingAuthorityURL* is indicated, the field *professionItems* of *ProfessionInfo* SHOULD contain only registered titles. If the field *professionOIDs* exists, it has to contain the OIDs of the professions listed in *professionItems* in the same order. In general, the field *professionInfos* SHOULD contain only one entry, unless the admissions that are to be listed are logically connected (e.g. they have been issued under the same admission number).

6) In P1.T23.[9] add

9	**id-ad-caIssuers** OBEJCT IDENTIFIER ::= {id-ad 2}	an OID for the case, when the referenced information lists CAs that have issued certificates for the issuer of this certificate.	+-	+-	4.2.2.1		[2]

3 Corrigenda to Part 2: PKI Management

1) In P2.T2#2.2 remove

2.2	*digestAlgorithms*	Collection (including zero) of message digest algorithm identifiers	RFC 2630 RFC 2315	5.1 9.1	++		++CA	++EE	~~OID: 1.3.14.3.2.26~~	[3]

2) In P2.T2.[3] remove

[3]	~~This OID identifies the SHA-1 hash algorithm, which shall be supported by compliant components. The support for other hash algorithms is OPTIONAL.~~ For permitted hash algorithm identifiers refer to P6.S2.1 (Cryptographic Algorithms) of this ISIS-MTT specification.

4 Corrigenda to Part 3: Message Formats

3) In P3.T5#8 replace

8	*signingCertificate* *id-aa-signingCertificate* {1 2 840 113549 1 9 16 212}	Sequence of certificate identifiers starting with the certificate of the signer	RFC 2634	5.4	+-		+-	+-	The *issuerSerial* field of the *ESSCertID* within *SigningCertificate* MUST not be empty.	[2], [5]

by

8	*signingCertificate* *id-aa-signingCertificate* {1 2 840 113549 1 9 16 2 12}	Sequence of certificate identifiers starting with the certificate of the signer	RFC 2634	5.4	+-		+-	+-	The *issuerSerial* field of the *ESSCertID* within *SigningCertificate* MUST not be empty.	[2], [5]

4) In P3.2.2 add

- *Content-Type* including the parameters *protocol*, *micalg (sha1, sha256, sha512, md5* or *unknown)*, and *boundary*,

5) In P3.T2#2 remove

2	*digestAlgorithms*	Collection (including zero) of message digest algorithm identifiers	RFC 3369	5.1	++		++	++	~~OID: 1.3.14.3.2.26~~	[3]

6) In P3.T2.[3] replace

[3]	~~This OID identifies the SHA-1 hash algorithm, which shall be supported by compliant components. The support for other hash algorithms is optional.~~

by

[3]	For permitted hash algorithm identifiers refer to P6.T1 (One-Way Hash Functions) of this ISIS-MTT specification..

7) In P3.T4#3 remove

3	*digestAlgorithm*	Identification of the signers hash algorithm	RFC 3369	5.3	++		++	++	~~OID: 1.3.14.3.2.26~~	[3]

8) In P3.T4.[3] replace

[3]	~~This OID identifies the SHA-1 hash algorithm, which shall be supported by compliant components. The support for other hash algorithms is optional.~~ The value provided in this field SHALL be contained in the *SignedData.digestAlgorithms* field (see T2.#2).

by

[3]	The value provided in this field SHALL be contained in the *SignedData.digestAlgorithms* field (see T2.#2). For permitted hash algorithm identifiers refer to P6.T1 (One-Way Hash Functions) of this ISIS-MTT specification.

5 Corrigenda to Part 4: Operational Protocols

1) In P4.T6.[1] replace

[1]	**ISIS-MTT PROFILE:** ~~The hash values in *certID* MUST be built using SHA-1. Processing components (typically the responder) MUST support SHA-1 and SHOULD support RIPEMD160 and MD5.~~

by

[1]	**ISIS-MTT PROFILE:** The hash functions to use for *certID* are defined in Table 1 of Part 6.

6 Corrigenda to Part 5: Certificate Path Validation

1) In P5.T6#3 add

3	`bool crlIsIndirect;`
	`if( cdp.IsEmpty() )`
	`  crlIsIndirect = false;`
	`else if( cdp.ContainsCrlIssuer() )`
	`  crlIsIndirect = true;`
	`else`
	`  crlIsIndirect = false;`
		In this step, it will be determined, whether the required CRL is an indirect one.
		If a *CRLDistributionPoints* extension in the certificate contains CRL access information and any of the CDPs contains the *crlIssuer* field, an indirect CRL is to be used.
		ISIS-MTT PROFILE: The support of indirect CRLs is RECOMMENDED.	

2) In P5.T5#4-6 replace

4	`if( tbvCert.AuthorityAccessInfoPresentAndContainsOcspUrl() )`
	`{`
	`  return CheckStatusViaOcsp( tbvCert,`
	`  refTime,`
	`  initialPolicySet,`
	`  trustedCerts,`
	`  trustedCrls );`
		This step is OPTIONAL. Actual implementations MAY or MAY NOT choose to support OCSP. If so and the certificate contains OCSP access info in the *AuthorityAccessInfo* extension, the revocation status will be checked using OCSP.
		It may furthermore be advantageous, to check first for an appropriate, locally available CRL, before using an on-line service.	

5	`else`
	`  return CheckStatusUsingCRL( tbvCert,`
	`  issCert,`
	`  tbvCerts,`
	`  refTime,`
	`  initialPolicySet,`
	`  trustedCerts,`
	`  trustedCrls );`
		The revocation status will be investigated using CRLs.	

6	`else`
	`  return false;`
	`}`
		Status could not be checked, because no directory access information was available.	

by

4	`if( tbvCert.AuthorityAccessInfoPresentAndContainsOcspUrl() )`
	`{`
	`  return CheckStatusViaOcsp( tbvCert,`
	`  refTime,`
	`  initialPolicySet,`
	`  trustedCerts,`
	`  trustedCrls );`
	`else`
		This step is OPTIONAL. Actual implementations MAY or MAY NOT choose to support OCSP. If so and the certificate contains OCSP access info in the *AuthorityAccessInfo* extension, the revocation status will be checked using OCSP.
		It may furthermore be advantageous, to check first for an appropriate, locally available CRL, before using an on-line service.	

5	~~`else`~~
	`  return CheckStatusUsingCRL( tbvCert,`
	`  issCert,`
	`  tbvCerts,`
	`  refTime,`
	`  initialPolicySet,`
	`  trustedCerts,`
	`  trustedCrls );`
	`}`
		The revocation status will be investigated using CRLs.	

~~6~~	~~`else`~~ ~~`  return false;`~~ ~~`}`~~	~~Status could not be checked, because no directory access information was available.~~	

7 Corrigenda to Part 6: Cryptographic Algorithms

1) In P6.T1#2 replace

2	SHA-256	one-way hash function	[XML_ENC] [FIPS-180-2]		n. a.	~~+~~	+	http://www.w3.org/2001/04/xmlenc#sha256	[~~3~~, 4]
3	SHA-512	one-way hash function	[XML_ENC] [FIPS-180-2]		n. a.	~~+~~	+	http://www.w3.org/2001/04/xmlenc#sha512	[~~3~~,4]

by

2	SHA-256	one-way hash function	[RFC 4055] [XML_ENC] [FIPS-180-2]		n. a.	+	+	OID: 2.16.840.1.101.3.4.2.1 http://www.w3.org/2001/04/xmlenc#sha256	[1, 4]
3	SHA-512	one-way hash function	[RFC 4055] [XML_ENC] [FIPS-180-2]		n. a.	+	+	OID: 2.16.840.1.101.3.4.2.3 http://www.w3.org/2001/04/xmlenc#sha512	[~~1~~,4]

2) In P6.T1.[1] add

[1]	**ISIS-MTT PROFILE:** SHA-1 is the preferred one-way hash function. This requirement is conformant with the PKIX and the XML_DSIG documents. SHA-1 is defined in [FIPS 180-1] and [ISO/IEC 10118-3]. In cases where SHA-1 will not be used due to security considerations, the preferred one-way hash function is SHA-256.

3) In P6.T2#1 to P6.T2#7 add and remove

1	sha-1WithRSAEncryption	RSA signature algorithm	[RFC3279] [RFC 2633] [FIPS 180-1] [ISO/IEC 10118-3]	2.2.1 2.2	+-	++	++	OID: 1.2.840.113549.1.1.5 http://www.w3.org/2000/09/xmldsig#rsa-sha1	[1,3,4,6] [7]

			[XML_DSIG]					
2	sha256WithRSAEncryption	RSA signature algorithm	[RFC 4055] [FIPS-180-2] [RFC 4051]		n. a.	+-	+	OID: 1.2.840.113549.1.1.11 http://www.w3.org/2001/04/xmldsig-more#rsa-sha256	[1,3,4,6] [7]
3	sha512WithRSAEncryption	RSA signature algorithm	[RFC 4055] [FIPS-180-2] [RFC 4051]		n. a.	+-	+	OID: 1.2.840.113549.1.1.13 http://www.w3.org/2001/04/xmldsig-more#rsa-sha512	[1,3,4,6] [7]
4	rsaSignatureWithRipemd160	RSA signature algorithm	[RIPEMD-160] [ISO/IEC 10118-3] [OSCI]		n. a.	-	+	OID: 1.3.36.3.3.1.2 http://www.w3.org/2001/04/xmlenc#ripemd160	[2,3,6] [7]
5	md2-WithRSAEncryption	RSA signature algorithm	[RFC3279] [RFC 2633]	2.2.1 2.2	+-	--	--	OID: 1.2.840.113549.1.1.2	[1,3,4,6]
6	md5WithRSAEncryption	RSA signature algorithm	[RFC3279] [RFC 2633]	2.2.1 2.2	+-	--	+-	OID: 1.2.840.113549.1.1.4	[1,3,4,6]
7	dsa-with-sha1	DSA signature algorithm	[RFC3279] [RFC2633] [FIPS 186-2] [XML_DSIG]	2.2.2 2.2	++	+- ++	++ ++	OID: 1.2.840.10040.4.3 http://www.w3.org/2000/09/xmldsig#dsa-sha1	[5] [7,8]
8	ecdsa-with-SHA1	ECDSA signature algorithm	[RFC3279] [X9.62]	2.2.3	+-	+-	+-	OID: 1.2.840.10045.4.1	

4) In P6.T2.[1] add

[1]	The PKIX documents do not make any recommendation which of the RSA signature algorithms (md2withRSAEncryption, md5withRSAEncryption, sha-1WithRSAEncryption) should be preferred. **ISIS-MTT PROFILE:** sha-1WithRSAEncryption is the preferred signature algorithm. In cases where sha1WithRSAEncryption will not be used due to security considerations, the preferred signature algorithm is sha256WithRSAEncryption.

5) In P6 section References add

[RFC 4051] D. Eastlake 3rd: Additional XML Security Uniform Resource Identifiers (URIs), April 2005

[RFC 4055] J. Schaad, B. Kaliski and R. Housley: Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, June 2005

8 Corrigenda to Part 7: Cryptographic Token Interface

Currently no corrigenda.

9 Corrigenda to Part 8: XML Profile

1) In References replace

[XML_EXCAN] W3C: Exclusive XML Canonicalization 1.0, 18 July 2002
~~http://www.w3.org/TR/2001/REC-xml-c14n-20010315~~

by

[XML_EXCAN] W3C: Exclusive XML Canonicalization 1.0, 18 July 2002
http://www.w3c.org/TR/2002/REC-xml-exc-c14n-20020718

10 Corrigenda to Optional Profile: SigG-Profile

1) In SigP.1.2 (3) add

(3) a flat, 3-layer certification hierarchy for accredited CAs: a governmental agency at the top level (responsible for policies, accreditation and subsequent supervision), certification service providers at the middle level (providing CA services for end entities, but not permitted to issue certificates for other CAs) and end entities at the bottom.

2) In SigP.6 append

SigG-conforming applications that support revocation checking by CRL as alternative to OCSP MUST be able to provess indirect CRLs.

In the context of SigG it is possible, that the DName of a CRL-issuer registered in the *CRLDistributionPoints* extension of a certificate must be changed. In this case it is possible, that the CRL for a certificate is signed by a different CRL-issuer than the registered one in the *CRLDistributionPoints* extension. If a client conforming to this profile (and optional a non-SigG client) downloads the CRL from the CDP URI and encounters this situation, it SHOULD check if the (valid, see also P1.T12.[1]) CRL-issuer, which signed the CRL, can be validated to the same root CA as the certificate being checked. If this is true, then the CRL SHOULD be considered as if it were signed by the original CRL-issuer.

This provision is an extension of the algorithm specified in Section 2.3 of Part 5, in particular step #4 of the *CheckStatusUsingCRL*() function in P5.T6. The modification of this step is given in Table 15, using the same tabular form and notation as in Part 5.

Table 15: CheckStatusUsingCRL()

#	PSEUDO-CODE	COMMENTS	NOTES

1	`Name crlIssuerDName;`
	`if( crlIsIndirect )`
	`  crlIssuerDName = cdp.crlIssuer.GetDirectoryName();`
	`else`
	`  crlIssuerDName = tbvCert.GetIssuerDName();`
		The DName of the CRL-issuer is determined.
		ISIS-MTT PROFILE: Note that the CDP MUST contain the DName of the issuer of each indirect CRL (P1.T22.#5 & [5]). For indirect CRLs, other CRL-issuer DNames SHOULD also be acceptable, provided there is a matching CRL-signing certificate that can be validated to the same root CA as *tbvCert*.	

Non-SigG-conforming applications MAY adopt this behaviour when evaluating indirect CRLs.

11 Corrigenda to Optional Profile: Optional Enhancements to the SigG-Profile

Currently no corrigenda.