Network Working Group
Internet-Draft
Intended status: Informational
Expires: June 9, 2014

S. Leontiev
P. Smirnov
CRYPTO-PRO
A. Chelpanov
InfoTeCS

December 6, 2013

Using GOST 28147-89, GOST R 34.10, and GOST R 34.11 Algorithms for XML Security
draft-chudov-cryptopro-cpxmldsig-09

Abstract

This document specifies how to use Russian national cryptographic standards GOST 28147-89, GOST R 34.10 and GOST R 34.11 with XML Signatures, XML Encryption, WS-SecureConversation, WS-SecurityPolicy and WS-Trust. A number of Uniform Resource Identifiers (URIs) and XML elements are defined.

The contents of this document is technically equivalent to its TC26 ROSSTANDART specification.

This specification is maintained by TC26 ROSSTANDART and further updates are available at: http://www.tc26.ru/.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on June 9, 2014.

Copyright Notice

Table of Contents

1. Introduction

This document specifies how to use GOST R 34.10 digital signatures and public keys, GOST R 34.11 hash, GOST 28147-89 encryption algorithms with XML Signatures [XMLDSIG], XML Encryption [XMLENC-CORE], WS-SecureConversation [WS-SECURECONVERSATION], WS-SecurityPolicy [WS-SECURITYPOLICY] and WS-Trust [WS-TRUST].

This document uses both XML Schema ([XML-SCHEMA-1], [XML-SCHEMA-2]) (normative) and DTD [XML] (informational) to specify the corresponding XML structures.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT","SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [KEYWORDS].

2. GOST Cryptographic Algorithms

Algorithms GOST R 34.10-2001, GOST R 34.11-94 and GOST 28147-89 have been developed by Russian Federal Agency of Governmental Communication and Information (FAGCI) and "All-Russian Scientific and Research Institute of Standardization". They are described in [GOSTR341001], [GOSTR341194] ([GOST3431004] and [GOST3431195]) and [GOST28147]. RECOMMENDED parameters for those algorithms are described in [CPALGS].

3. Version and Namespaces

This specification makes no provision for an explicit version number in the syntax. If a future version is needed, it will use a different namespace.

The XML namespace [XML-NS] URI [RFC3986] that MUST be used by implementations of this (dated) specification is:

```
urn:ietf:params:xml:ns:cpxmlsec
```

The following external XML namespaces are used in this specification (without line breaks; the choice of any namespace prefix is arbitrary and not semantically significant):

```
http://www.w3.org/2000/09/xmldsig#
   Prefix:
      dsig
```

Specification:
[XMLDSIG]

http://www.w3.org/2001/04/xmlenc#
Prefix:
xenc
Specification:
[XMLENC-CORE]

http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702
Prefix:
sp
Specification:
[WS-SECURITYPOLICY]

http://www.w3.org/ns/ws-policy
Prefix:
wsp
Specification:
[WS-POLICY]

http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512
Prefix:
wsc
Specification:
[WS-SECURECONVERSATION]

http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-secext-1.0.xsd
Prefix:
wsse
Specification:
[WS-SECURITY]

http://docs.oasis-open.org/ws-sx/ws-trust/200512/
Prefix:
wst
Specification:
[WS-TRUST]

In the remaining sections of this document elements in the external namespaces are marked as such by using the namespace prefixes defined above.

4. XML Schema Preamble and DTD Replacement

4.1. XML Schema Preamble

The subsequent preamble is to be used with the XML Schema definitions given in the remaining sections of this document.

```
<xs:schema
  xmlns:cpxmlsec="urn:ietf:params:xml:ns:cpxmlsec"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:sp=
  "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
  targetNamespace="urn:ietf:params:xml:ns:cpxmlsec"
  elementFormDefault="qualified"
  version="0.4">
```

4.2. DTD Replacement

In order to include GOST XML-signature syntax, the following definition of the entity Key.ANY SHOULD replace the one in [XMLDSIG]:

```
<!ENTITY % KeyValue.ANY '| cpxmlsec:GOSTKeyValue'>
```

5. Object Identifiers Representation

Object Identifiers (OIDs) are included in XML by the corresponding URN value as defined in [URNOID].

The subsequent type is to be used to define algorithm parameters by OIDs:

```
<xs:simpleType name="ObjectIdentifierType">
  <xs:restriction base="xs:anyURI">
    <xs:pattern value=
      "urn:oid:(([0-1]\.[1-3]?\d)|(2\.\d+))(\.\d+)*" />
  </xs:restriction>
</xs:simpleType>
```

6. Specifying GOST within XML Signature and XML Encryption

This section specifies the details of how to use GOST algorithms with XML Signature Syntax and Processing [XMLDSIG] and XML Encryption Syntax and Processing [XMLENC-CORE]. It relies heavily on syntaxes and namespaces defined in [XMLDSIG] and [XMLENC-CORE].

6.1. GOST R 34.11-94 Algorithm in DigestMethod

The identifier for the GOST R 34.11-94 digest algorithm is:

urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr3411

The dsig:DigestMethod node may contain a child node cpxmlsec: ParametersR3411 specifying parameters for GOST R 34.11-94 algorithm. cpxmlsec:ParametersR3411 node contains one OID specified in section 8.2 [CPALGS]. If cpxmlsec:ParametersR3411 node is missing, the application should infer algorithm parameters from other sources.

If the application omits cpxmlsec:ParametersR3411 node, it SHOULD use parameters defined by id-GostR3411-94-CryptoProParamSet (see Section 11.2 of [CPALGS]).

Schema Definition:

```
<xs:element name="ParametersR3411"
            type="cpxmlsec:ObjectIdentifierType"/>
```

DTD Definition:

```
<!ELEMENT ParametersR3411 (#PCDATA) >
```

An example of a GOST R 34.11-94 dsig:DigestMethod node is:

```
<dsig:DigestMethod dsig:Algorithm=
    "urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr3411">
  <!-- id-GostR3411-94-CryptoProParamSet -->
  <cpxmlsec:ParametersR3411>urn:oid:1.2.643.2.2.30.1<
  /cpxmlsec:ParametersR3411>
</dsig:DigestMethod>
```

A GOST R 34.11-94 digest is a 256-bit string. The content of the dsig:DigestValue element shall be the base64 [RFC4648] encoding of this bit string viewed as a 32-octet octet stream.

6.2. GOST R 34.11-2012 Algorithm with 256-bit output in DigestMethod

The identifier for the GOST R 34.11-2012 digest algorithm with 256-bit output is:

urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34112012-256

An example of a GOST R 34.11-2012 with 256-bit output dsig: DigestMethod node is:

```
<dsig:DigestMethod dsig:Algorithm=
    "urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34112012-256" />
```

A GOST R 34.11-2012 digest in this case is a 256-bit string. The content of the dsig:DigestValue element shall be the base64 [RFC4648] encoding of this bit string viewed as a 32-octet octet stream.

6.3. GOST R 34.11-2012 Algorithm with 512-bit output in DigestMethod

The identifier for the GOST R 34.11-2012 digest algorithm with 512-bit output is:

urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34112012-512

An example of a GOST R 34.11-2012 with 512-bit output dsig: DigestMethod node is:

```
<dsig:DigestMethod dsig:Algorithm=
    "urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34112012-512" />
```

A GOST R 34.11-2012 digest in this case is a 512-bit string. The content of the dsig:DigestValue element shall be the base64 [RFC4648] encoding of this bit string viewed as a 64-octet octet stream.

6.4. GOST R 34.11-94 HMAC Algorithm in SignatureMethod

GOST R 34.11-94 can also be used in HMAC [HMAC] as described in section 6.3.1 of [XMLDSIG]. Identifier:

urn:ietf:params:xml:ns:cpxmlsec:algorithms:hmac-gostr3411

The dsig:SignatureMethod node may contain a child node cpxmlsec: ParametersR3411 specifying parameters for GOST R 34.11-94 algorithm. cpxmlsec:ParametersR3411 node syntax and processing in this case are equivalent to the ones in dsig:DigestMethod case.

An example of a GOST R 34.11-94 HMAC disg:SignatureMethod node is:

```
  <dsig:SignatureMethod dsig:Algorithm=
      "urn:ietf:params:xml:ns:cpxmlsec:algorithms:hmac-gostr3411">
    <!-- id-GostR3411-94-CryptoProParamSet -->
    <cpxmlsec:ParametersR3411>urn:oid:1.2.643.2.2.30.1<
    /cpxmlsec:ParametersR3411>
```

```
  </dsig:SignatureMethod>
```

The output of the GOST R 34.11-94 HMAC algorithm is ultimately the output of the GOST R 34.11-94 digest algorithm. This value shall be base64 [RFC4648] encoded for the dsig:SignatureValue in the same straightforward fashion as the output of the digest algorithm in Section 6.1.

6.5. GOST R 34.10-2001 Algorithm in SignatureMethod

The input to the GOST R 34.10-2001 algorithm is the canonicalized representation of the dsig:SignedInfo element as specified in Section 3 of [XMLDSIG].

The identifier for the GOST R 34.10-2001 signature algorithm is (without line break):

```
urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34102001-gostr3411
```

An example of a GOST R 34.10-2001 dsig:SignatureMethod node is (without line break in attribute value):

```
<dsig:SignatureMethod dsig:Algorithm=
"urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34102001-
gostr3411" />
```

GOST R 34.10-2001 signature is a 64-octet value as described in section 2.2.2 of [CPPK]. The content of the dsig:SignatureValue element shall be the base64 [RFC4648] encoding of this value.

6.6. GOST R 34.10-2012 Algorithm in SignatureMethod

The input to the GOST R 34.10-2012 algorithm is the canonicalized representation of the dsig:SignedInfo element as specified in Section 3 of [XMLDSIG].

The identifiers for the GOST R 34.10-2012 signature algorithm are (without line breaks):

```
urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34102012-
gostr34112012-256
urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34102012-
gostr34112012-512
```

Both identifiers refer to GOST R 34.11-2012 as digest algorithm. The first one denotes that the 256-bit output version of that algorithm is used, the second one corresponds to 512-bit output.

An example of a GOST R 34.10-2012 dsig:SignatureMethod node is (without line break in attribute value):

```
<dsig:SignatureMethod dsig:Algorithm=
"urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34102012-
gostr34112012-256" />
```

6.7. GOST R 34.10-2001 Public Key in KeyValue

6.7.1. Key Value Root Element

GOST R 34.10-2001 public key can be transmitted in cpxmlsec: GOSTKeyValue node. It is included in dsig:KeyValue node just like dsig:RSAKeyValue or xenc:DHKeyValue.

cpxmlsec:GOSTKeyValue node consists of an optional child node cpxmlsec:PublicKeyParameters and a mandatory child node cpxmlsec: PublicKey. If cpxmlsec:PublicKeyParameters node is missing, the application should infer parameters from other sources.

Schema Definition:

```
<xs:element name="GOSTKeyValue"
            type="cpxmlsec:KeyValueType"/>

<xs:complexType name="KeyValueType">
  <xs:sequence>
    <xs:element name="PublicKeyParameters"
                type="cpxmlsec:PublicKeyParametersType"
                minOccurs="0"/>
    <xs:element name="PublicKey" type="xs:base64Binary"/>
  </xs:sequence>
</xs:complexType>
```

DTD Definition:

```
<!ELEMENT GOSTKeyValue (
            PublicKeyParameters?, PublicKey) >
<!ELEMENT PublicKey (#PCDATA) >
```

If the application omits cpxmlsec:PublicKeyParameters node, it SHOULD use parameters identified by DefaultPublicKeyParameters.

DefaultPublicKeyParameters:

```
<cpxmlsec:PublicKeyParameters>
  <!-- id-GostR3410-2001-CryptoPro-A-ParamSet -->
  <cpxmlsec:publicKeyParamSet>urn:oid:1.2.643.2.2.35.1<
  /cpxmlsec:publicKeyParamSet>
  <!-- id-GostR3411-94-CryptoProParamSet -->
  <cpxmlsec:digestParamSet>urn:oid:1.2.643.2.2.30.1</
  cpxmlsec:digestParamSet>
  <!-- id-Gost28147-89-CryptoPro-A-ParamSet -->
  <cpxmlsec:encryptionParamSet>urn:oid:1.2.643.2.2.31.1</
  cpxmlsec:encryptionParamSet>
</cpxmlsec:PublicKeyParameters>
```

6.7.2. Public Key Parameters

cpxmlsec:PublicKeyParameters node contains three OIDs: cpxmlsec: publicKeyParamSet, cpxmlsec:digestParamSet and optional cpxmlsec: encryptionParamSet. Parameter values corresponding to these OIDs can be found in [CPALGS].

Schema Definition:

```
<xs:complexType name="PublicKeyParametersType">
   <xs:sequence>
      <xs:element name="publicKeyParamSet"
                  type="cpxmlsec:ObjectIdentifierType"/>
      <xs:element name="digestParamSet"
                  type="cpxmlsec:ObjectIdentifierType"/>
      <xs:element name="encryptionParamSet"
                  type="cpxmlsec:ObjectIdentifierType"
                  minOccurs="0"/>
   </xs:sequence>
</xs:complexType>
```

DTD Definition:

```
<!ELEMENT PublicKeyParameters (
              publicKeyParamSet, digestParamSet,
              encryptionParamSet?) >
<!ELEMENT publicKeyParamSet (#PCDATA) >
<!ELEMENT digestParamSet (#PCDATA) >
<!ELEMENT encryptionParamSet (#PCDATA) >
```

6.8. GOST R 34.10-2001-based Key Agreement Algorithm in AgreementMethod

Key agreement algorithm based on GOST R 34.10-2001 public keys (see Section 5 of [CPALGS]) involves the derivation of shared secret information using keys from the sender and recipient.

The identifier for the key agreement algorithm based on GOST R 34.10-2001 is:

urn:ietf:params:xml:ns:cpxmlsec:algorithms:agree-gost2001

An example of a GOST R 34.10-2001-based key agreement AgreementMethod node is:

```
<xenc:AgreementMethod xenc:Algorithm=
    "urn:ietf:params:xml:ns:cpxmlsec:algorithms:agree-gost2001">
  <xenc:KA-Nonce>...</xenc:KA-Nonce>
  <xenc:OriginatorKeyInfo>
    <dsig:X509Data><dsig:X509Certificate>
      ...
    </dsig:X509Certificate></dsig:X509Data>
  </xenc:OriginatorKeyInfo>
  <xenc:RecipientKeyInfo><dsig:KeyValue>
    ...
  </dsig:KeyValue></xenc:RecipientKeyInfo>
</xenc:AgreementMethod>
```

The shared keying material for algorithm based on GOST R 34.10-2001 needed will be calculated as a result of function VKO GOST R 34.10-2001 (see Section 5.2 of [CPALGS]), which generates GOST KEK using two GOST R 34.10-2001 keypairs and UKM. xenc:KA-Nonce node of xenc: AgreementMethod contains base64 encoded 64-bits value of UKM, if UKM is used.

6.9. GOST R 34.10-2001-based Key Transport Algorithm in EncryptionMethod

The key transport algorithm based on VKO GOST R 34.10-2001, specified in [CPALGS], is public key encryption algorithms, that MUST be used for key encryption/decryption only.

The identifier for the key transport algorithm based on VKO GOST R 34.10-2001 is:

urn:ietf:params:xml:ns:cpxmlsec:algorithms:transport-gost2001

An example of a VKO GOST R 34.10-2001-based key transport EncryptedKey node is:

```
<xenc:EncryptedKey>
  <xenc:EncryptionMethod xenc:Algorithm=
"urn:ietf:params:xml:ns:cpxmlsec:algorithms:transport-gost2001" />
  <dsig:KeyInfo>
    <dsig:X509Data><dsig:X509Certificate>
      ...
    </dsig:X509Certificate></dsig:X509Data>
  </dsig:KeyInfo>
  <xenc:CipherData>
      <xenc:CipherValue>...</xenc:CipherValue>
  </xenc:CipherData>
</xenc:EncryptedKey>
```

The CipherValue for such encrypted key is the base64 encoding of the [X.208-88] DER encoding of a GostR3410-KeyTransport structure (see section 4.2.1 of [CPCMS]).

6.10. GOST 28147-89 Algorithm in EncryptionMethod

The identifier for the GOST 28147-89 symmetric encryption algorithm is:

urn:ietf:params:xml:ns:cpxmlsec:algorithms:gost28147

The xenc:EncryptionMethod node may contain a child node cpxmlsec: Parameters28147 specifying parameters for GOST 28147-89 algorithm. cpxmlsec:Parameters28147 specifies the set of corresponding Gost28147-89-ParamSetParameters (see Section 8.1 of [CPALGS]). Encryption mode is specified by mode parameter of Gost28147-89-ParamSetParameters structure. CFB and CNT modes are RECOMMENDED to use. If cpxmlsec:Parameters28147 node is missing, the application should infer algorithm parameters from other sources.

If the application omits cpxmlsec:Parameters28147 node, it SHOULD use parameters defined by id-Gost28147-89-CryptoPro-A-ParamSet (see Section of 10.2 [CPALGS]).

Schema Definition:

```
<xs:element name="Parameters28147"
            type="cpxmlsec:ObjectIdentifierType" />
```

DTD Definition:

```
<!ELEMENT Parameters28147 (#PCDATA) >
```

An example of a GOST 28147-89 xenc:EncryptionMethod node is:

```
<xenc:EncryptionMethod dsig:Algorithm=
    "urn:ietf:params:xml:ns:cpxmlsec:algorithms:gost28147">
  <!-- id-Gost28147-89-CryptoPro-A-ParamSet -->
  <cpxmlsec:Parameters28147>urn:oid:1.2.643.2.2.31.1<
  /cpxmlsec:Parameters28147>
</xenc:EncryptionMethod>
```

256-bit key, 64-bit Initialization Vector (IV), and optional parameters are used in GOST 28147-89 encryption algorithm. The resulting cipher text is prefixed by the IV. If included in XML output, it is then base64 encoded.

6.11. GOST 28147-89 authenticated encryption in EncryptionMethod

The identifier for the GOST 28147-89 authenticated encryption algorithm is:

urn:ietf:params:xml:ns:cpxmlsec:algorithms:gost28147aead

The xenc:EncryptionMethod node may contain a child node cpxmlsec: Parameters28147 specifying parameters for GOST 28147-89 algorithm.

If the application omits cpxmlsec:Parameters28147 node, it SHOULD use parameters defined by id-tc26-gost-28147-param-Z.

An example of a GOST 28147-89 AEAD xenc:EncryptionMethod node is:

```
<xenc:EncryptionMethod dsig:Algorithm=
    "urn:ietf:params:xml:ns:cpxmlsec:algorithms:gost28147aead">
  <!-- id-tc26-gost-28147-param-Z -->
  <cpxmlsec:Parameters28147>urn:oid:1.2.643.7.1.2.5.1.1<
  /cpxmlsec:Parameters28147>
</xenc:EncryptionMethod>
```

256-bit key, 64-bit Initialization Vector (IV), and optional parameters are used in GOST 28147-89 authenticated encryption algorithm. The resulting cipher text is prefixed by the IV and suffixed by the MAC. Standard XML encryption padding, id-Gost28147-89-CryptoPro-KeyMeshing, imitovstavka and CNT mode should be used

during encryption. If included in XML output, it is then base64 encoded.

6.12. Symmetric Key Wrap

Symmetric Key Wrap algorithms considered in this section are shared secret key encryption algorithms that MUST be used for symmetric keys encryption/decryption only.

6.12.1. GOST 28147-89 Key Wrap in EncryptionMethod

The GOST 28147-89 Key Wrap algorithm wraps (encrypts) a key (the wrapped key, WK) under a GOST 28147-89 Key Wrap (specified in sections 6.1, 6.2 of [CPALGS]).

Note: This algorithm MUST NOT be used without key agreement algorithm, because such WK is constant for every wrapping-encrypting pair. Encrypting many different keys with the same constant WK may reveal that WK. The only key agreement algorithm possible to use with GOST 28147-89 Key Wrap defined by this specification is a GOST R 34.10-2001-based key agreement (see Section 6.8).

The identifier for the GOST 28147-89 Key Wrap algorithm is:

urn:ietf:params:xml:ns:cpxmlsec:algorithms:kw-gost

The CipherValue for such wrapped key is the base64 encoding of the [X.208-88] DER encoding of a GostR3410-KeyWrap structure.

ASN.1 structure:

```
GostR3410-KeyWrap ::=
    SEQUENCE {
        encryptedKey Gost28147-89-EncryptedKey,
        encryptedParameters Gost28147-89-KeyWrapParameters
     }
```

An example of a GOST 28147-89 Key Wrap EncryptedData node is:

```
<xenc:EncryptedData>
  <xenc:EncryptionMethod dsig:Algorithm=
    "urn:ietf:params:xml:ns:cpxmlsec:algorithms:gost28147" />
  <dsig:KeyInfo>
    <xenc:EncryptedKey>
      <xenc:EncryptionMethod xenc:Algorithm=
"urn:ietf:params:xml:ns:cpxmlsec:algorithms:kw-gost" />
        <dsig:KeyInfo>
          <xenc:AgreementMethod xenc:Algorithm=
"urn:ietf:params:xml:ns:cpxmlsec:algorithms:agree-gost2001">
            <xenc:KA-Nonce>...</xenc:KA-Nonce>
            <xenc:OriginatorKeyInfo>
              <dsig:X509Data><dsig:X509Certificate>
                ...
              </dsig:X509Certificate></dsig:X509Data>
            </xenc:OriginatorKeyInfo>
            <xenc:RecipientKeyInfo><dsig:KeyValue>
              ...
            </dsig:KeyValue></xenc:RecipientKeyInfo>
          </xenc:AgreementMethod>
        </dsig:KeyInfo>
      <xenc:CipherData>
          <xenc:CipherValue>...</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedKey>
  </dsig:KeyInfo>
  <xenc:CipherData>
      <xenc:CipherValue>...</xenc:CipherValue>
  </xenc:CipherData>
</xend:EncryptedData>
```

Gost28147-89-KeyWrapParameters is described in section 4.1.1 of [CPCMS]. The xenc:KA-Nonce node value of the xenc:AgreementMethod node MUST be used as ukm.

The resulting wrapped key (WK) is placed in the Gost28147-89-EncryptedKey encryptedKey field, its mac (CEK_MAC) is placed in the Gost28147-89-EncryptedKey macKey field. ukm field of Gost28147-89-KeyWrapParameters MUST be absent.

6.12.2. CryptoPro Key Wrap in EncryptionMethod

The CryptoPro Key Wrap algorithm wraps (encrypts) a key (wrapped key, WK) under a CryptoPro Key Wrap (specified in sections 6.3, 6.4 of [CPALGS]).

The identifier for the CryptoPro Key Wrap algorithms is:

urn:ietf:params:xml:ns:cpxmlsec:algorithms:kw-cp

The CipherValue for such wrapped key is the base64 encoding of the [X.208-88] DER encoding of a GostR3410-KeyWrap structure (see Section 6.12.1).

An example of a CryptoPro Key Wrap EncryptedData node is:

```
  <xenc:EncryptedData>
    <xenc:EncryptionMethod dsig:Algorithm=
      "urn:ietf:params:xml:ns:cpxmlsec:algorithms:gost28147" />
    <dsig:KeyInfo>
      <xenc:EncryptedKey>
        <xenc:EncryptionMethod xenc:Algorithm=
    "urn:ietf:params:xml:ns:cpxmlsec:algorithms:kw-cp" />
          <dsig:KeyInfo>
            <dsig:KeyName>John Smith</dsig:KeyName>
          </dsig:KeyInfo>
        <xenc:CipherData>
            <xenc:CipherValue>...</xenc:CipherValue>
        </xenc:CipherData>
      </xenc:EncryptedKey>
    </dsig:KeyInfo>
    <xenc:CipherData>
        <xenc:CipherValue>...</xenc:CipherValue>
    </xenc:CipherData>
  </xend:EncryptedData>
```

The resulting wrapped key (WK) is placed in the Gost28147-89-EncryptedKey encryptedKey field, its mac (CEK_MAC) is placed in the Gost28147-89-EncryptedKey macKey field.

If CryptoPro Key Wrap algorithm is combined with Key Agreement Algorithm, the xenc:KA-Nonce node value of the xenc:AgreementMethod node MUST be used as ukm. ukm field of Gost28147-89-KeyWrapParameters type must be absent.

Note: The only key agreement algorithm possible to use with CryptoPro Key Wrap defined by this specification is a GOST R 34.10-2001-based key agreement (see Section 6.8).

If CryptoPro Key Wrap algorithm is not combined with Key Agreement Algorithm, ukm field of Gost28147-89-KeyWrapParameters type MUST be present.

7. Specifying GOST within WS-*

This section specifies the details of how to use GOST algorithms with WS-SecureConversation [WS-SECURECONVERSATION], WS-SecurityPolicy [WS-SECURITYPOLICY] and WS-Trust [WS-TRUST].

7.1. GOST Algorithm Suite for WS-SecurityPolicy

This specification defines a new possible value for an [Algorithm Suite] property of a Security Binding (see section 6.1 of [WS-SECURITYPOLICY]). The new value is BasicGost.

BasicGost Algorithm Suite defines the following values for operations and properties (without line breaks in URIs):

```
[Sym Sig]
   urn:ietf:params:xml:ns:cpxmlsec:algorithms:hmac-gostr3411
[Asym Sig]
   urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34102001-
   gostr3411
[Dig]
   urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr3411
[Enc]
   urn:ietf:params:xml:ns:cpxmlsec:algorithms:gost28147
[Sym KW]
   urn:ietf:params:xml:ns:cpxmlsec:algorithms:kw-cp
[Asym KW]
   urn:ietf:params:xml:ns:cpxmlsec:algorithms:transport-gost2001
[Comp Key]
   urn:ietf:params:xml:ns:cpxmlsec:algorithms:dk-p-gostr3411
[Enc KD]
   urn:ietf:params:xml:ns:cpxmlsec:algorithms:dk-p-gostr3411
[Sig KD]
   urn:ietf:params:xml:ns:cpxmlsec:algorithms:dk-p-gostr3411
[Min SKL]
   256
[Max SKL]
   256
[Min AKL]
   512
[Max AKL]
   512
```

Note: For definition of [Comp Key], [Enc KD] and [Sig KD] algorithm see Section 7.2

To indicate a requirement to use GOST Algorithm Suite defined above conforming implementations MUST place cpxmlsec:BasicGost node in sp: AlgorithmSuite Assertion (see section 7.1 of [WS-SECURITYPOLICY]).

Schema Definition:

```
<xs:element name="BasicGost"
            type="sp:QNameAssertionType"/>
```

DTD Definition:

```
<!ELEMENT BasicGost EMPTY >
```

An example of a GOST Algorithm Suite in sp:AlgorithmSuite Assertion is:

```
<sp:AlgorithmSuite>
  <wsp:Policy>
    <cpxmlsec:BasicGost/>
  </wsp:Policy>
</sp:AlgorithmSuite>
```

7.2. GOST Key Derivation Algorithm for WS-SecureConversation

This specification defines a new possible value for an Algorithm attribute of a wsc:DerivedKeyToken node (see section 7 of [WS-SECURECONVERSATION]).

The new key derivation algorithm identifier is:

```
urn:ietf:params:xml:ns:cpxmlsec:algorithms:dk-p-gostr3411
```

An example of a GOST Key Derivation Algorithm in wsc:DerivedKeyToken node is:

```
<wsc:DerivedKeyToken Algorithm=
  "urn:ietf:params:xml:ns:cpxmlsec:algorithms:dk-p-gostr3411">
  <wsse:SecurityTokenReference>...</wsse:SecurityTokenReference>
  <wsc:Nonce>...</wsc:Nonce>
</wsc:DerivedKeyToken>
```

GOST Key Derivation Algorithm uses a pseudo-random function P_GOSTR3411 (see section 4 of [CPALGS]) to derive keys just like a P_SHA-1 function is used in [WS-SECURECONVERSATION] (see section 7).

7.3. GOST Computed Key Mechanism for WS-Trust

This specification defines a new possible value for a wst:ComputedKey node (see section 4.4.4 of [WS-TRUST]).

The new computed key mechanism identifier is:

urn:ietf:params:xml:ns:cpxmlsec:algorithms:ck-p-gostr3411

An example of a GOST Computed Key Mechanism in wst:ComputedKey node (without line breaks) is:

```
<wst:ComputedKey>
  urn:ietf:params:xml:ns:cpxmlsec:algorithms:ck-p-gostr3411
</wst:ComputedKey>
```

GOST Computed Key Mechanism uses a pseudo-random function P_GOSTR3411 (see section 4 of [CPALGS]) to compute a key just like a P_SHA-1 function is used in [WS-TRUST] (see section 4.4.4). It is REQUIRED that EntREQ and EntRES are strings of length 256 bits.

7.4. Using WS-Trust for TLS Handshake with GOST Algorithm Suite

This specification defines how to use WS-Trust ([WS-TRUST]) to perform TLS Handshake (see [TLS]) and establish secure session for GOST Algorithm Suite.

WS-Trust can be used to do TLS Handshake as specified in [WS-TRUST-TLS]. The outcome of the protocol under discussion is a new session key issued using a secure session established by TLS Handshake. Issued session key is intended to secure further communication by means of WS-Security ([WS-SECURITY]).

If application is required to use GOST Algorithm Suite after performing TLS Handshake by WS-Trust it MUST use one of GOST 28147-89 Cipher Suites for TLS (see [draft.CPTLS]).

The main flow of TLS Negotiation over WS-Trust defined in this specification complies with [WS-TRUST-TLS], but there are a few differences specified below that MUST be obeyed.

The paragraph R4305 (see section 4.3 of [WS-TRUST-TLS]) MUST be replaced with the following text:

> The responder is responsible for issuing the key associated with the TLSNego session. If the initiator requested properties for the generated key (e.g. key size) in the initial RST message, the generated key SHOULD match those requirements. The issued key

MUST be communicated back to the initiator using the wst: RequestedProofToken element and MUST be protected using CryptoPro Key Wrap algorithm (see section 6.3 of [CPALGS]) where server_write_key (see section 6.3 of [TLS]) is a wrapping key. Wrapped key is contained in the <xenc:CipherData><xenc: CipherValue>...</xenc:CipherValue></xenc:CipherData> elements of the xenc:EncryptedKey.

GOST R 34.11-94 and P_GOSTR3411 algorithms MUST be used instead of SHA1 and PSHA1 algorithms correspondingly to compute authenticator (see section 4.9 of [WS-TRUST-TLS]).

8. Security Considerations

Conforming applications MUST use unique values for ukm and iv. Recipients MAY verify that ukm and iv specified by the sender are unique.

Applications SHOULD verify signature values, subject public keys and algorithm parameters to conform to [GOSTR341001], standard before using them.

Cryptographic algorithm parameters affect algorithm strength. Using parameters not listed in [CPALGS] is NOT RECOMMENDED (see the Security Considerations section of [CPALGS]).

Using the same key for signature and key derivation is NOT RECOMMENDED.

It is NOT RECOMMENDED to use XML encryption without XML signature or HMAC.

9. IANA Considerations

This document uses URNs to describe XML namespaces and XML schemata conforming to a registry mechanism described in [RFC3688]. IANA has registered two URI assignments.

9.1. URN Sub-Namespace Registration for urn:ietf:params:xml:ns:cpxmlsec

URI: urn:ietf:params:xml:ns:cpxmlsec

Registrant Contact:
Mikhail V. Pavlov

CRYPTO-PRO, Ltd.
16/5, Suschevskij val
Moscow, 127018
Russia
Phone: +7 (495) 780 4820
Fax: +7 (495) 660 2330
Email: pav@CryptoPro.ru
URI: http://www.CryptoPro.ru

XML: None. Namespace URIs do not represent an XML specification.

9.2. Schema Registration

URI: urn:ietf:params:xml:schema:cpxmlsec

Registrant Contact:
Mikhail V. Pavlov
CRYPTO-PRO, Ltd.
16/5, Suschevskij val
Moscow, 127018
Russia
Phone: +7 (495) 780 4820
Fax: +7 (495) 660 2330
Email: pav@CryptoPro.ru
URI: http://www.CryptoPro.ru

XML: The XML can be found in Appendix A.

10. References

10.1. Normative references

[CPALGS] Popov, V., Kurepkin, I., and S. Leontiev, "Additional Cryptographic Algorithms for Use with GOST 28147-89, GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms", RFC 4357, January 2006.

[CPCMS] Leontiev, S. and G. Chudov, "Using the GOST 28147-89, GOST R 34.11-94, GOST R 34.10-94, and GOST R 34.10-2001 Algorithms with Cryptographic Message Syntax (CMS)", RFC 4490, May 2006.

[CPPK] Leontiev, S. and D. Shefanovski, "Using the GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms with the Internet X.509 Public Key Infrastructure Certificate and CRL Profile", RFC 4491, May 2006.

[GOST28147]
Government Committee of the USSR for Standards, "Cryptographic Protection for Data Processing System, Gosudarstvennyi Standard of USSR (In Russian)", GOST 28147-89, 1989.

[GOST3431004]
Council for Standardization, Metrology and Certification of the Commonwealth of Independence States (EASC), Minsk, "Information technology. Cryptographic Data Security. Formation and verification processes of (electronic) digital signature based on Asymmetric Cryptographic Algorithm (In Russian)", GOST 34.310-2004, 2004.

[GOST3431195]
Council for Standardization, Metrology and Certification of the Commonwealth of Independence States (EASC), Minsk, "Information technology. Cryptographic Data Security. Cashing function (In Russian)", GOST 34.311-95, 1995.

[GOSTR341001]
Government Committee of the Russia for Standards, "Information technology. Cryptographic Data Security.Signature and verification processes of [electronic] digital signature, Gosudarstvennyi Standard of Russian Federation (In Russian)", GOST R 34.10-2001, 2001.

[GOSTR341194]
Government Committee of the Russia for Standards, "Information technology. Cryptographic Data Security. Hashing function, Gosudarstvennyi Standard of Russian Federation (In Russian)", GOST R 34.11-94, 1994.

[HMAC] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[KEYWORDS]
Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[TLS] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[WS-POLICY] Vedamuthu, A., Orchard, D., Hirsch, F., Hondo, M., Yendluri, P., Boubez, T., and . Yalinalp, "Web Services Policy 1.5 - Framework", W3C REC-ws-policy, September 2007, <http://www.w3.org/TR/ws-policy/>.

[WS-SECURECONVERSATION] Lawrence, K. and C. Kaler, "WS-SecureConversation 1.3", OASIS Standard ws-secureconversation-1.3-os, March 2007, <http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.html>.

[WS-SECURITY] Lawrence, K. and C. Kaler, "Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)", OASIS Standard wss-v1.1-spec-os-SOAPMessageSecurity, Febraury 2006, <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>.

[WS-SECURITYPOLICY] Lawrence, K. and C. Kaler, "WS-SecurityPolicy 1.2", OASIS Standard ws-securitypolicy-1.2-spec-os, July 2007, <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.html>.

[WS-TRUST] Lawrence, K. and C. Kaler, "WS-Trust 1.3", OASIS Standard ws-trust-1.3-os, March 2007, <http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html>.

[WS-TRUST-TLS] Alexander, J., Della-Libera, G., Gajjala, V., Gavrylyuk, K., Kaler, C., McIntosh, M., Nadalin, A., Rich, B., and T. Vishwanath, "Application Note: Using WS-Trust for TLS Handshake", September 2007, <http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-trust/WSTrustForTLS-final.pdf>.

[X.208-88] International International Telephone and Telegraph Consultative Committee, "Specification of Abstract Syntax

Notation One (ASN.1)", CCITT Recommendation X.208, November 1988.

[XML-NS] Bray, T., Hollander, D., Layman, A., and R. Tobin, "Namespaces in XML (Second Edition)", W3C REC-xml-names, August 2006, <http://www.w3.org/TR/REC-xml-names-20060816>.

[XML-SCHEMA-1] Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition", W3C REC-xmlschema-1, October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>.

[XML-SCHEMA-2] Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", W3C REC-xmlschema-2, October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>.

[XMLDSIG] Eastlake, D., Reagle, J., and D. Solo, "(Extensible Markup Language) XML-Signature Syntax and Processing", RFC 3275, March 2002.

[XMLENC-CORE] Eastlake, D. and J. Reagle , "XML Encryption Syntax and Processing", W3C Candidate Recommendation xmlenc-core, August 2002, <http://www.w3.org/TR/xmlenc-core/>.

[draft.CPTLS] Afanasiev, A., Nikishin, N., Izotov, B., Minaeva, E., Murugov, S., Ustinov, I., Erkin, A., Chudov, G., and S. Leontiev, "GOST 28147-89 Cipher Suites for Transport Layer Security (TLS)", draft-chudov-cryptopro-cptls-04 (work in progress), December 2008.

10.2. Informative references

[RFC4134] Hoffman, P., "Examples of S/MIME Messages", RFC 4134, July 2005.

[URNOID] Mealling, M., "A URN Namespace of Object Identifiers", RFC 3061, February 2001.

[XML] Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fourth Edition)", W3C REC-xml, August 2006, <http://www.w3.org/TR/2006/REC-xml-20060816>.

Appendix A. Aggregate XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- Declare helper entity to avoid overrunning right margin of RFC
     text while importing WS-SecurityPolicy schema.-->
<!DOCTYPE schema [
    <!ENTITY ws-securitypolicyuri
        "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
]>

<xs:schema
    xmlns:cpxmlsec="urn:ietf:params:xml:ns:cpxmlsec"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:sp=
    "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
    targetNamespace="urn:ietf:params:xml:ns:cpxmlsec"
    elementFormDefault="qualified"
    version="0.4">

  <xs:import namespace=
    "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
             schemaLocation=
    "&ws-securitypolicyuri;/ws-securitypolicy-1.2.xsd" />

  <xs:simpleType name="ObjectIdentifierType">
    <xs:restriction base="xs:anyURI">
      <xs:pattern
        value="urn:oid:(([0-1]\.[1-3]?\d)|(2\.\d+))(\.\d+)*" />
    </xs:restriction>
  </xs:simpleType>

  <xs:element name="ParametersR3411"
              type="cpxmlsec:ObjectIdentifierType" />

  <xs:element name="GOSTKeyValue" type="cpxmlsec:KeyValueType" />

  <xs:complexType name="KeyValueType">
    <xs:sequence>
      <xs:element name="PublicKeyParameters"
                  type="cpxmlsec:PublicKeyParametersType"
                  minOccurs="0"/>
      <xs:element name="PublicKey" type="xs:base64Binary" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="PublicKeyParametersType">
```

```
        <xs:sequence>
           <xs:element name="publicKeyParamSet"
                       type="cpxmlsec:ObjectIdentifierType" />
           <xs:element name="digestParamSet"
                       type="cpxmlsec:ObjectIdentifierType" />
           <xs:element name="encryptionParamSet"
                       type="cpxmlsec:ObjectIdentifierType"
                       minOccurs="0" />
        </xs:sequence>
     </xs:complexType>

     <xs:element name="Parameters28147"
                 type="cpxmlsec:ObjectIdentifierType" />

     <xs:element name="BasicGost"
                 type="sp:QNameAssertionType"/>

   </xs:schema>
```

Appendix B. Aggregate DTD

```
    <!ELEMENT GOSTKeyValue (
                  PublicKeyParameters?, PublicKey) >
    <!ELEMENT PublicKey (#PCDATA) >
    <!ELEMENT PublicKeyParameters (
                  publicKeyParamSet, digestParamSet,
                  encryptionParamSet?) >
    <!ELEMENT publicKeyParamSet (#PCDATA) >
    <!ELEMENT digestParamSet (#PCDATA) >
    <!ELEMENT encryptionParamSet (#PCDATA) >
    <!ELEMENT Parameters28147 (#PCDATA) >
    <!ELEMENT ParametersR3411 (#PCDATA) >
    <!ELEMENT BasicGost EMPTY >
```

Appendix C. Examples

Examples here are stored in the same format as the examples in [RFC4134] and can be extracted using the same program.

If you want to extract without the program, copy all the lines between the "|>" and "|<" markers, remove any page breaks, and remove the "|" in the first column of each line. The result is a valid Base64 blob that can be processed by any Base64 decoder.

C.1. Signed document

This sample contain the signed XML document using the sample certificate from Section 4.2 of [CPPK].

```
|>XmlDocSigned2001.xml
|PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0idXRmLTgiPz48Q3J5cHRvUHJv
|WE1MIFNpZ25lZD0idHJ1ZSI+SGVyZSBpcyBzb21lIGRhdGEgdG8gc2lnbi48U2ln
|bmF0dXJlIHhtbG5zPSJodHRwOi8vd3d3LnczLm9yZy8yMDAwLzA5L3htbGRzaWcj
|Ij48U2lnbmVkSW5mbz48Q2Fub25pY2FsaXphdGlvbk1ldGhvZCBBbGdvcml0aG09
|Imh0dHA6Ly93d3cudzMub3JnL1RSLzIwMDEvUkVDLXhtbC1jMTRuLTIwMDEwMzE1
|IiAvPjxTaWduYXR1cmVNZXRob2QgQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9y
|Zy8yMDAxLzA0L3htbGRzaWctbW9yZSNnb3N0cjM0MTAyMDAxLWdvc3RyMzQxMSIg
|Lz48UmVmZXJlbmNlIFVSST0iIj48VHJhbnNmb3Jtcz48VHJhbnNmb3JtIEFsZ29y
|aXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMC8wOS94bWxkc2lnI2VudmVsb3Bl
|ZC1zaWduYXR1cmUiIC8+PC9UcmFuc2Zvcm1zPjxEaWdlc3RNZXRob2QgQWxnb3Jp
|dGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbGRzaWctbW9yZSNnb3N0
|cjM0MTEiIC8+PERpZ2VzdFZhbHVlPi9Kd3RRc3Z5NWsvUjBWZUx6ZG0ySWlqUEJ0
|U0o1cEpSalQ5RlVRSEV5VGc9PC9EaWdlc3RWYWx1ZT48L1JlZmVyZW5jZT48L1Np
|Z25lZEluZm8+PFNpZ25hdHVyZVZhbHVlPkZjYjNxNGlCdmRmZ1lvN245NUdhUUN1
|ZDkxWVA3dzhvVjAzUjZ6a1JEZGxjK0RuQ2MwcjlNc0E1YS9iaFlDeVdQZC9jRVU4
|K3FZRnJ5SmJjaXJ5d0hBPT08L1NpZ25hdHVyZVZhbHVlPjxLZXlJbmZvPjxYNTA5
|RGF0YT48WDUwOUNlcnRpZmljYXRlPk1JSUIwRENDQVg4Q0VDdjF4aDdDRWIwWHg5
|elVZbWEwTGlFd0NBWUdLb1VEQWdJRE1HMHhIekFkQmdOVkJBTU1Ga2R2YzNSU016
|UXhNQzB5TURBeElHVjRZVzF3YkdVeEVqQVFCZ05WQkFvTUNVTnllWEIwYjFCeWJ6
|RUxNQWtHQTFVRUJoTUNVbFV4S1RBbkJna3Foa2lHOXcwQkNRRVdHa2R2YzNSU016
|UXhNQzB5TURBeFFHVjRZVzF3YkdVdVkyOXRNQjRYRFRBMU1EZ3hOakUwTVRneU1G
|b1hEVEUxTURneE5qRTBNVGd5TUZvd2JURWZNQjBHQTFVRUF3d1dSMjl6ZEZJek5E
|RXdMVEl3TURFZ1pYaGhiWEJzWlRFU01CQUdBMVVFQ2d3SlEzSjVjSFJ2VUhKdk1R
|c3dDUVlEVlFRR0V3SlNWVEVwTUNjR0NTcUdTSWIzRFFFSkFSWWFSMjl6ZEZJek5E
|RXdMVEl3TURGQVpYaGhiWEJzWlM1amIyMHdZekFjQmdZcWhRTUNBaE13RWdZSEtv
|VURBZ0lrQUFZSEtvVURBZ0llQVFOREFBUkFoSlZvZFdBQ0drQjFDTTBUakRHSkxQ
|M2xCUU42UTF6MGJTc1A1MDh5ZmxlUDY4d1d1WldJQTlDYWZJV3VEK1NONnFhN2Zs
|Ykh5N0RmRDJhOHl1b2FZREFJQmdZcWhRTUNBZ01EUVFBOEw4a0pSTGNucWV5bjFl
|bjdVMjNTdzZwa2ZFUXUzdTB4RmtWUHZGUS8zY0hlRjI2TkcreHh0WlB6M1RhVFZY
|ZG9pWWtYWWlEMDJyRXgxYlVjTTk3aTwvWDUwOUNlcnRpZmljYXRlPjwvWDUwOURh
|dGE+PC9LZXlJbmZvPjwvU2lnbmF0dXJlPjwvQ3J5cHRvUHJvWE1MPg==
|<XmlDocSigned2001.xml
```

Appendix D. Acknowledgments

The authors wish to thank:

Microsoft Corporation Russia for provided information about company products and solutions, and also for technical consulting in PKI.

Our colleague Grigorij S. Chudov for writing the first version of this document.

Authors' Addresses

Serguei E. Leontiev
"CRYPTO-PRO", LLC
18, Suschevsky Val str.
Moscow 127018
Russian Federation

Phone: +7 (916) 686 10 81
Fax: +74957804820
Email: lse@cryptopro.ru
URI: http://www.cryptopro.ru

Pavel V. Smirnov
"CRYPTO-PRO", LLC
18, Suschevsky Val str.
Moscow 127018
Russian Federation

Phone: +7 (495) 780 4820
Fax: +74957804820
Email: spv@CryptoPro.ru
URI: http://www.CryptoPro.ru

Aleksandr V. Chelpanov
JSC "InfoTeCS"
build 1, 1/23, Staryj Petrovsko-Razumovsij pr.
Moscow 127287
Russia

Phone: +7 (495) 737-6192
Fax: +7 (495) 737-7278
Email: Aleksandr.Chelpanov@infotecs.ru