Digital Signatures for the v1.0 Internet Open Trading Protocol (IOTP)

Status of this Memo

Copyright Notice

Abstract

   A syntax and procedures are described for the computation and
   verification of digital signatures for use within Version 1.0 of the
   Internet Open Trading Protocol (IOTP).

Acknowledgment

Table of Contents

1. Introduction

   The Internet Open Trading Protocol (IOTP) provides a payment system
   independent interoperable framework for Internet commerce as
   documented in [RFC 2801]. All IOTP messages are XML documents. XML,
   the Extensible Markup Language [XML], is a syntactical standard
   promulgated by the World Wide Web Consortium. XML is intended
   primarily for structuring data exchanged and served over the World
   Wide Web.

   Although IOTP assumes that any payment system used with it provides
   its own security, there are numerous cases where IOTP requires
   authentication and integrity services for portions of the XML
   messages it specifies.

2. Objective and Requirements

   This document covers how digital signatures may be used with XML
   documents to provide authentication and tamper-proof protocol
   messages specifically for Version 1.0 of the IOTP protocol. The
   reader should recognize that an effort towards general XML digital
   signatures exists but is unlikely to produce its final result in time
   for IOTP Version 1.0. Future versions of IOTP will probably adopt by
   reference the results of this general XML digital signature effort.

   The objective of this document is to propose syntax and procedures
   for the computation and verification of digital signatures applicable
   to Version 1.0 IOTP protocol messages, providing for:

   -- Authentication of IOTP transactions
   -- Provide a means by which an IOTP message may be made "tamper-
      proof", or detection of tampering is made evident
   -- Describe a set of available digest and signature algorithms at
      least one of which is mandatory to implement for interoperability
   -- Easily integrate within the IOTP 1.0 Specification
   -- Provide lightweight signatures with minimal redundancy
   -- Allow signed portions of IOTP message to be "forwarded" to another
      trading roles with different signature algorithms than the
      original recipient

3. Signature Basics

3.1 Signature Element

   This specification consists primarily of the definition of an XML
   element known as the Signature element. This element consists of two
   sub-elements. The first one is a set of authenticated attributes,
   known as the signature Manifest, which comprises such things as a

   unique reference to the resources being authenticated and an
   indication of the keying material and algorithms being used. The
   second sub-element consists of the digital signature value.

```
   <Signature>
           <Manifest>
                   (resource information block)
                   (originator information block)
                   (recipient information block)
                   (other attributes)
                   (signature algorithms information block)
           </Manifest>
           <Value encoding 'encoding scheme'>
                   (encoded signature value)
           <Value>
   </Signature>
```

   The digital signature is not computed directly from the pieces of
   information to be authenticated. Instead, the digital signature is
   computed from a set of authenticated attributes (the Manifest), which
   include references to, and a digests of, those pieces of information.

   The authentication is therefore "indirect".

3.2 Digest Element

   The Digest element consists of a unique and unambiguous reference to
   the XML resources being authenticated. It is constructed of a locator
   and the digest value data itself. The Digest algorithm is referred to
   indirectly via a DigestAlgorithmRef, so that Digest algorithms may be
   shared by multiple resources.

```
   <Digest DigestAlgorithmRef='D.1'>
       <Locator href='resource locator'/>
       <Value>
            (Digest value)
       </Value>
   </Digest>
```

   The resource locator is implemented as a simple XML Link [XLink].
   This not only provides a unique addressing scheme for internal and
   external resources, but also facilitates authentication of composite
   documents.

3.3 Originator and Recipient Information Elements

   The purpose of the Originator and Recipient information elements is
   to provide identification and keying material for these respective
   parties.

   <OriginatorInfo>
       (identification information block)
       (keying material information block)
   </OriginatorInfo>

   <RecipientInfo>
       (identification information block)
       (keying material information block)
   </RecipientInfo>

   The actual content of these two elements depends on the
   authentication scheme being used and the existence or non-existence
   of a prior relationship between the parties. In some circumstances,
   it may be quite difficult to distinguish between identification and
   keying material information. A unique reference to a digital
   certificate provides for both. This may also stand true for an
   account number when a prior relationship exists between the parties.

   The Originator information element is mandatory. Depending on the
   existence or non-existence of a prior relationship with the
   recipient, this block either refers to a public credential such as a
   digital certificate or displays a unique identifier known by the
   recipient.

   The Recipient information element may be used when a document
   contains multiple signature information blocks, each being intended
   for a particular recipient. A unique reference in the Recipient
   information block helps the recipients identify their respective
   Signature information block.

   The Recipient information element may also be used when determination
   of the authentication key consists of a combination of keying
   material provided by both parties. This would be the case, for
   example, when establishing a key by means of Diffie Hellman
   [Schneier] Key Exchange algorithm.

3.4 Algorithm Element

   The Algorithm element is a generalized place to put any type of
   algorithm used within the signed IOTP message. The Algorithm may be a
   Signature algorithm or a Digest algorithm. Each algorithm contains
   parameters specific to the algorithm used.

```
      <Algorithm type='digest' ID='12'>
          (algorithm information block)
      </Algorithm>
```

   Algorithms are required to contain an ID which allows for indirect
   reference to them from other places in the XML signature.

4. Detailed Signature Syntax

4.1 Uniform Resource Names

   To prevent potential name conflicts in the definition of the numerous
   type qualifiers considered herein, this specification uses Uniform
   Resource Names [RFC 2141].

4.2 IotpSignatures

   The IotpSignatures element is the top-level element in an IOTP
   signature block. It consists of a collection of Signature elements,
   and an optional set of Certificates.

```
   <!ELEMENT IotpSignatures (Signature+, Certificate*) >
   <!ATTLIST IotpSignatures
           ID               ID              #IMPLIED >
```

   Content Description

   Signature: A collection of Signature elements.

   Certificate: Zero or more certificates used for signing

   Attributes Description

   ID: Element identifier that may be used to reference the entire
   Signature element from a Resource element when implementing
   endorsement.

4.3 Signature Component

4.3.1 Signature

   The Signature element constitutes the majority of this specification.
   It is comprised of two sub-elements. The first one is a set of
   attributes, known as the Manifest, which actually constitutes the
   authenticated part of the document. The second sub-element consists
   of the signature value or values.

The Value element contained within the Signature element is the
encoded form of the signature of the Manifest element, and thus
provides the verification of the Manifest.

The process for generating the signed value is a multi-step process,
involving a canonicalization algorithm, a digest algorithm, and a
signature algorithm.

First, the Manifest is canonicalized with an algorithm specified
within the Algorithm element of the Manifest. The canonicalized form
removes any inconsistencies in white space introduced by XML parsing
engines.

This canonicalized form is then converted into a digest form which
uniquely identifies the canonicalized document. Any slight
modification in the original document will result in a very different
digest value.

Finally, the digest is then signed using a public/symmetric key
algorithm which digitally stamps the digest (with the certificate of
the signer if available). The final signed digest is the value which
is stored within the Signature's Value element.

```
<!ELEMENT Signature (Manifest, Value+) >
<!ATTLIST Signature
       ID               ID              #IMPLIED >
```

Content Description

Manifest: A set of attributes that actually constitutes the
authenticated part of the document.

Value: One or more encodings of signature values. Multiple values
allow for a multiple algorithms to be supported within a single
signature component.

Attributes Description

ID: Element identifier that may be used to reference the Signature
element from a Resource element when implementing endorsement.

4.3.2 Manifest

The Manifest element consists of a collection of attributes that
specify such things as references to the resources being
authenticated and an indication of the keying material and algorithms
to be used.

```
<!ELEMENT Manifest
        (        Algorithm+,
                 Digest+,
                 Attribute*,
                 OriginatorInfo,
                 RecipientInfo+,
        )
<!ATTLIST Manifest
        LocatorHRefBase          CDATA          #IMPLIED
>
```

Content Description

Algorithm: A list of algorithms used for signing, digest computation, and canonicalization.

Digest: A list of digests of resources to be authentication and signed.

Attribute: Optional element that consists of a collection of complementary attributes to be authenticated.

OriginatorInfo: Element that provides identification and keying material information related to the originator.

RecipientInfo: Optional element that provides identification and keying material information related to the recipient.

Attributes Description

LocatorHrefBase: The LocatorHrefBase provides a similar construct to the HTML HREFBASE attribute and implicitly sets all relative URL references within the Manifest to be relative to the HrefBase. For example, the IOTP Manifest may contain:

<Manifest LocatorHrefBase='iotp:<globally-unique-tid>'>

And subsequent Locators may be:

<Locator href='C.9'>

An implementation should concatenate the two locator references with "#" to create the entire URL. See definition of the Locator attribute on the Digest element for more detail.

4.3.3 Algorithm

   This specification uses an Algorithm data type which indicates many
   different types of algoirithms. The Algorithm element allows for
   specification of sub-algorithms as parameters of the primary
   algorithm. This is performed via a parameter within the algorithm
   that provides a reference to another Algorithm. An example of this is
   shown in the Parameter section.

   <!ELEMENT Algorithm (Parameter*) >
   <!ATTLIST Algorithm
           ID   ID   #REQUIRED
           type   (digest|signature)   #IMPLIED
           name   NMTOKEN   #REQUIRED >

   Content Description

   Parameter: The contents of an Algorithm element consists of an
   optional collection of Parameter elements which are specified on a
   per algorithm basis.

   Attributes Description

   ID: The ID of the algorithm is used by the Digest and RecipientInfo
   to refer to the signing or digest algorithm used.

   type: The type of algorithm, either a digest or signature. This is
   implied by the element to which the algorithm is referred. That is,
   if the DigestAlgorithmRef refers to an algorithm, it is implicit by
   reference that the targeted algorithm is a digest.

   name: The type of the algorithm expressed as a Uniform Resource
   Name.

4.3.4 Digest

   The Digest element consists of the fingerprint of a given resource.
   This element is constructed of two sub-elements. This first one
   indicates the algorithm to be used for computation of the
   fingerprint. The second element consists of the fingerprint value.

   <!ELEMENT Digest (Locator, Value) >
   <!ATTLIST Digest
         DigestAlgorithmRef   IDREF   #REQUIRED
   >

   Content Description

   Locator: Contains a "HREF" or URL Locator for the resources to be
   fingerprinted. For use within IOTP a "scheme" with the value "iotp"
   may be used with the following structure:

      'iotp:<globally-unique-tid>#<id-value>'.

   This should be interpreted as referring to an element with an ID
   attribute that matches <id-value> in any IOTP Message that has a
   TransRefBlk Block with an IotpTransId that matches <globally-unique-
   tid>.

   If the LocatorHrefBase attribute is set on the Manifest element of
   which this Digest element is a child, then concatenate the value of
   the LocatorHrefBase attribute with the value of the Locator attribute
   before identifying the element that is being referred to.

   If the LocatorHrefBase attribute is omitted, <globally-unique-tid>
   should be interpreted as the current IotpTransId, which is included
   in the IOTP message which contains the Manifest component.

   Value: Encoding of the fingerprint value.

   Attributes Description

   DigestAlgorithmRef: ID Reference of algorithm used for computation of
   the digest.

4.3.5 Attribute

   The Attribute element consists of a complementary piece of
   information, which shall be included in the authenticated part of the
   document. This element has been defined primarily for enabling some
   level of customization in the signature element. This is the area
   where a specific IOTP implementation may include custom attributes
   which must be authenticated directly. An Attribute element consists
   of a value, a type, and a criticality.

   At this time, no IOTP specific attributes are specified.

```
   <!ELEMENT Attribute ANY >
   <!ATTLIST Attribute
         type                NMTOKEN            #REQUIRED
         critical         ( true | false )      #REQUIRED
   >
```

   Content Description

   ANY: The actual value of an attribute depends solely upon its type.

   Attributes Description

   type: Type of the attribute.

   critical: Boolean value that indicates if the attribute is critical
   (true) or not (false). A recipient shall reject a signature that
   contains a critical attribute that he does not recognize. However, an
   unrecognized non-critical attribute may be ignored.

4.3.6 OriginatorInfo

   The OriginatorInfo element is used for providing identification and
   keying material information for the originator.

   <!ELEMENT OriginatorInfo ANY >
   <!ATTLIST OriginatorInfo
         OriginatorRef   NMTOKEN   #IMPLIED
   >

   Content Description

   ANY: Identification and keying material information may consist of
   ANY construct. Such a definition allows the adoption of
   application-specific schemes.

   Attributes Description

   OriginatorRef: A reference to the IOTP Org ID of the originating
   signer.

4.3.7 RecipientInfo

   The RecipientInfo element is used for providing identification and
   keying material information for the recipient. This element is used
   either for enabling recognition of a Signature element by a given
   recipient or when determination of the authentication key consists of
   the combination of keying material provided by both the recipient and
   the originator.

   The RecipientInfo attributes provide a centralized location where
   signatures, algorithms, and certificates intended for a particular
   recipient are specified.

The signature certificate reference ID MUST point to a certificate object.

```
<!ELEMENT RecipientInfo ANY >
<!ATTLIST RecipientInfo
        SignatureAlgorithmRef    IDREF         #REQUIRED
        SignatureValueRef        IDREF         #IMPLIED
        SignatureCertRef         IDREF         #IMPLIED
        RecipientRefs            NMTOKENS      #IMPLIED
>
```

Content Description

ANY: Identification and keying material information may consist of ANY construct.

Attributes Description

SignatureAlgorithmRef: A reference to the signature algorithm used to sign the SignatureValueRef intended for this recipient. The signature algorithm reference ID MUST point to a signature algorithm within the Manifest.

SignatureValueRef: A reference to the signature value for this recipient. The signature value reference ID MUST point to a value structure directly included within a Manifest. This reference can be omitted if the application can specify the digest value.

SignatureCertRef: A reference to the certificate used to sign the Value pointed to by the SignatureValueRef. This reference can be omitted if the application can identify the certificate.

RecipientRefs: A list of references to the IOTP Org ID of the recipients this signature is intended for.

4.3.8 KeyIdentifier

The key identifier element can identify the shared public/symmetric key identification between parties that benefit from a prior relationship. This element can be included in the ReceipientInfo Element.

```
<!ELEMENT KeyIdentifier EMPTY>
<!ATTLIST KeyIdentifier
  value              CDATA         #REQUIRED
  >
```

4.3.9 Parameter

   A Parameter element provides the value of a particular algorithm
   parameter, whose name and format have been specified for the
   algorithm considered.

   <!ELEMENT Parameter ANY >
   <!ATTLIST Parameter
           type   CDATA   #REQUIRED
   >

   For IOTP 1.0, the following parameter type is standardized:
   "AlgorithmRef".

   An AlgorithmRef contains an ID of a "sub-Algorithm" used when
   computing a sequence of algorithms. For example, a signature
   algorithm actually signs a digest algorithm. To specify a chain of
   algorithms used to compute a signature, AlgorithmRef parameter types
   are used in the following manner:

```
<Algorithm ID='A1' type='digest' name='urn:ibm-com:dom-hash'>
        <Parameter type='AlgorithmRef'>A2</Parameter>
</Algorithm>
<Algorithm ID='A2' type='digest' name='urn:nist-gov:sha1'>
</Algorithm>
<Algorithm ID='A3' type='signature' name='urn:rsasdi-com:rsa-encryption'>
        <Parameter type='AlgorithmRef'>A1</Parameter>
</Algorithm>
```

   Content Description

   ANY: The contents of a Parameter element consists of ANY valid
   construct, which is specified on a per algorithm per parameter basis.

   Attributes Description

   type: The type of the parameter expressed as a free form string,
   whose value is specified on a per algorithm basis.

4.4 Certificate Component

4.4.1 Certificate

   The Certificate element may be used for either providing the value of
   a digital certificate or specifying a location from where it may be
   retrieved.

```
<!ELEMENT Certificate
(        IssuerAndSerialNumber,
         ( Value | Locator ) )
>
<!ATTLIST Certificate
         ID              ID              #IMPLIED
         type            NMTOKEN         #REQUIRED >
```

Content Description

IssuerAndSerialNumber: Unique identifier of this certificate. This
element has been made mandatory is order to prevent unnecessary
decoding during validation of a certificate chain. This feature also
helps certificates caching, especially when the value is not directly
provided.

Value: Encoding of the certificate value. The actual value to be
encoded depends upon the type of the certificate.

Locator: XML link element that could be used for retrieving a copy of
the digital certificate. The actual value being returned by means of
this locator depends upon the security protocol being used.

Attributes Description

ID: Element identifier that may be used to reference the Certificate
element from a RecipientInfo element.

type: Type of the digital certificate. This attribute is specified as
a Universal Resource Name. Support for the X.509 version 3
certificate [X.509] is mandatory in this specification if the
Certificate element is used. The URN for such certificates is
"urn:X500:X509v3".

4.4.2 IssuerAndSerialNumber

The IssuerAndSerialNumber element identifies a certificate, and
thereby an entity and a public key, by the name of the certificate
issuer and an issuer-specific certificate identification.

```
<!ELEMENT IssuerAndSerialNumber EMPTY >
<!ATTLIST IssuerAndSerialNumber
         issuer          CDATA           #REQUIRED
         number          CDATA           #REQUIRED >
```

   Attributes Description

   issuer: Name of the issuing certification authority. See [RFC 2253]
   for RECOMMENDED syntax.

   number: Issuer-specific certificate identification.

4.5 Common Components

4.5.1 Value

   A value contains the "raw" data of a signature or digest algorithm,
   usually in a base-64 encoded form. See [RFC 2045] for algorithm used
   to base-64 encode data.

   <!ELEMENT Value (#PCDATA) >
   <!ATTLIST Value
           ID   ID   #IMPLIED
           encoding   (base64|none)   'base64'
   >

   Content Description

   PCDATA: Content value after adequate encoding.

   Attributes Description

   encoding: This attribute specifies the decoding scheme to be
   employed for recovering the original byte stream from the content of
   the element. This document recognizes the following two schemes:

   none: the content has not been subject to any particular encoding.
   This does not preclude however the use of native XML encoding such as
   CDATA section or XML escaping.

   base64: The content has been encoded by means of the base64 encoding
   scheme.

4.5.2 Locator

   The Locator element consists of simple XML link [XLink]. This
   element allows unambiguous reference to a resource or fragment of a
   resource.

   <!ELEMENT Locator EMPTY>
   <!ATTLIST Locator
           xml:link   CDATA   #FIXED   'simple'
           href   CDATA   #REQUIRED >

   Attributes Description

   xml:link: Required XML link attribute that specifies the nature of
   the link (simple in this case).

   href: Locator value that may contains either a URI [RFC 2396], a
   fragment identifier, or both.

5. Supported Algorithms

   The IOTP specification 1.0 requires the implementation of the DSA,
   DOM-HASH, SHA1, HMAC algorithms. Implementation of RSA is also
   recommended.

5.1 Digest Algorithms

   This specification contemplates two types of digest algorithms, both
   of which provide a digest string as a result:

   Surface string digest algorithms

   These algorithms do not have any particular knowledge about the
   content being digested and operate on the raw content value. Any
   changes in the surface string of a given content affect directly the
   value of the digest being produced.

   Canonical digest algorithms

   These algorithms have been tailored for a particular content type and
   produce a digest value that depends upon the core semantics of such
   content. Changes limited to the surface string of a given content do
   not affect the value of the digest being produced unless they affect
   the core semantic.

5.1.1 SHA1

   Surface string digest algorithm designed by NIST and NSA for use with
   the Digital Signature Standard. This algorithm produces a 160-bit
   hash value. This algorithm is documented in NIST FIPS Publication
   180-1 [SHA1].

   This algorithm does not require any parameter.

   The SHA1 URN used for this specification is "urn:nist-gov:sha1".

5.1.2 DOM-HASH

   XML canonical digest algorithm proposed by IBM Tokyo Research
   Laboratory. This algorithm operates on the DOM representation of the
   document and provides an unambiguous means for recursive computation
   of the hash value of the nodes that constitute the DOM tree [RFC
   2803]. This algorithm has many applications such as computation of
   digital signature and synchronization of DOM trees. However, because
   the hash value of an element is computed from the hash values of the
   inner elements, this algorithm is better adapted to small documents
   that do not require one-pass processing.

   As of today, this algorithm is limited to the contents of an XML
   document and, therefore, does not provide for authentication of the
   internal or external subset of the DTD.

   The DOM-HASH algorithm requires a single parameter, which shall
   include a surface string digest algorithm such as SHA1.

   The DOM-HASH URN used for this specification is "urn:ibm-com:dom-
   hash".

   The DOM-HASH uses a surface-string digest algorithm for computation
   of a fingerprint. The SHA1 is recommended in this specification.

   Example
   <Algorithm name='urn:fips:sha1' type='digest' ID='P.3'>
   </Algorithm>

   <Algorithm name='urn:ibm:dom-hash' type='digest' ID='P.5'>
   </Algorithm>

5.2 Signature Algorithms

   This specification uses the terminology of 'digital signature' for
   qualifying indifferently digital signature and message authentication
   codes. Thus, the signature algorithms contemplated herein include
   public key digital signature algorithms such as ECDSA and message
   authentication codes such as HMAC [RFC 2104].

5.2.1 DSA

   Public-key signature algorithm proposed by NIST for use with the
   Digital Signature Standard. This standard is documented in NIST FIPS
   Publication 186 [DSS] and ANSI X9.30 [X9.30].

The DSA algorithm requires a single parameter, which includes the
canonical digest algorithm to be used for computing the fingerprint
of the signature Manifest.

The DSA URN used in this specification is "urn:nist-gov:dsa".

The DSA uses a canonical or surface-string digest algorithm for
computation of the Manifest fingerprint. The DOM-HASH is recommended
in this specification.

Signature Value Encoding:

The output of this algorithm consists of a pair of integers usually
referred by the pair (r, s). The signature value shall consist of the
concatenation of two octet-streams that respectively result from the
octet-encoding of the values r and s. Integer to octet-stream
conversion shall be done according to PKCS#1 [RFC 2437] specification
with a k parameter equals to 20.

Example
```
<Algorithm name='urn:nist-gov:dsa' type='signature' ID='P.3'>
  <Parameter type='AlgorithmRef'>P.4</Parameter>
</Algorithm>
<Algorithm name='urn:ibm-com:dom-hash' type='digest' ID='P.4'>
  <Parameter type='AlgorithmRef'>P.5</Parameter>
</Algorithm>
<Algorithm name='urn:nist-gov:sha1' type='digest' ID='P.5'>
</Algorithm>
```

5.2.2 HMAC

Message Authentication Code proposed by H. Krawczyk et al., and
documented in [RFC 2104].

This specification adopts a scheme that differs a bit from the common
usage of this algorithm -- computation of the MAC is performed on the
hash of the contents being authenticated instead of the actual
contents. Thence, the actual signature value output by the algorithm
might be depicted as follows:

    SignatureValue = HMAC(SecretKey, H(Manifest))

This specification also considered HMAC output truncation such as
proposed by Preneel and van Oorschot. In their paper [PV] these two
researchers have shown some analytical advantages of truncating the
output of hash-based MAC functions. Such output truncation is also
considered in the RFC document.

HMAC requires three parameters. The first one consists of a canonical digest algorithm. The second one consists of a hash function. The last one is optional and specifies the length in bit of the truncated output. If this last parameter is absent, no truncation shall occur.

The HMAC URN used in this specification is "urn:ietf-org:hmac".

Canonical digest algorithm: Canonical or surface-string digest algorithm is to be used for computation of the Manifest fingerprint. The type of this parameter is set to "AlgorithmRef". The recommended value of this Parameter should be the ID reference for the Algorithm element DOM-HASH.

Hash-function: Hash function is to be used to compute the MAC value from the secret key and the fingerprint of the signature Manifest. The type of this parameter is set to "HashAlgorithmRef" and the value of this parameter should be set to the ID reference for the Algorithm element of SHA1.

Output-length: Length in bits of the truncated MAC value. The type of this parameter is set to "KeyLength" and the value of this parameter is set the length in bits of the truncated MAC value.

Signature Value Encoding:

The output of this algorithm can be assumed as a large integer value. The signature value shall consist of the octet-encoded value of this integer. Integer to octet-stream conversion shall be done according to PKCS#1 [RFC 2437] specification with a k parameter equals to ((Hlen +7) mod8), Mlen being the length in bits of the MAC value.

```
Example
<Algorithm name='urn:ietf-org:hmac' type='signature' ID='P.3'>
  <Parameter type='AlgorithmRef'>P.4</Parameter>
  <Parameter type='HashAlgorithmRef'>P.5</Parameter>
  <Parameter type='KeyLength'>128</Parameter>
</Algorithm>
<Algorithm name='urn:ibm-com:dom-hash' type='digest' ID='P.4'>
  <Parameter type='AlgorithmRef'>P.5</Parameter>
</Algorithm>
<Algorithm name='urn:nist-gov:sha1' type='digest' ID='P.5'>
</Algorithm>
```

5.2.3 RSA

   Public-key signature algorithm proposed by RSA Laboratories and
   documented in PKCS#1 [RFC 2437].

   This specification adopts the RSA encryption algorithm with padding
   block type 01. For computing the signature value, the signer shall
   first digest the signature Manifest and then encrypt the resulting
   digest with his private key.

   This signature algorithm requires a single parameter, which consists
   of the canonical digest algorithm to be used for computing the
   fingerprint of the signature Manifest.

   Specifications

   The RSA URN used in this specification is "urn:rsasdi-com:rsa-
   encription".

   The RSA uses a canonical or surface-string digest algorithm for
   computation of the Manifest fingerprint. The DOM-HASH is recommended
   in this specification.

   Signature Value Encoding:

   The output of this algorithm consists of single octet-stream. No
   further encoding is required.

   Example
   <Algorithm name='urn:rsasdi-com:rsa-encription'
                                    type='signature' ID='P.3'>
   </Algorithm>
   <Algorithm name='urn:ibm-com:dom-hash' type='digest' ID='P.4'>
   </Algorithm>
   <Algorithm name='urn:nist-gov:sha1' type='digest' ID='P.5'>
   </Algorithm>

5.2.4 ECDSA

   Public-key signature algorithm proposed independently by Neil Koblitz
   and Victor Miller. This algorithm is being proposed as an ANSI
   standard and is documented in ANSI X9.62 standard proposal [X9.62]
   and IEEE/P1363 standard draft proposal [IEEE P1363].

The ECDSA algorithm requires a single parameter, which consists of
the canonical digest algorithm to be used for computing the
fingerprint of the signature Manifest.

Specifications

The ECDSA URN used in this specification is "urn:ansi-org:ecdsa".

The ECDSA uses a canonical or surface-string digest algorithm for
computation of the Manifest fingerprint. The DOM-HASH [RFC 2803] is
recommended in this specification.

Signature Value Encoding:

The output of this algorithm consists of a pair of integers usually
referred by the pair (r, s). The signature value shall consist of the
concatenation of two octet-streams that respectively result from the
octet-encoding of the values r and s. Integer to octet-stream
conversion shall be done according to PKCS#1 [RFC 2437] specification
with a k parameter equals to 20.

Example
```
<Algorithm name='urn:ansi-org:ecdsa' type='signature' ID='P.3'>
  <Parameter type='AlgorithmRef'>P.4</Parameter>
</Algorithm>
<Algorithm name='urn:ibm-com:dom-hash' type='digest' ID='P.4'>
  <Parameter type='AlgorithmRef'>P.5</Parameter>
</Algorithm>
<Algorithm name='urn:nist-gov:sha1' type='digest' ID='P.5'>
</Algorithm>
```

6. Examples

The following is an example signed IOTP message:

```
<IotpMessage>
  <TransRefBlk ID='M.1'>
    <TransId
        ID='M.2'
        version='1.0'
        IotpTransID='19990809215923@www.iotp.org'
        IotpTransType='BaselinePurchase'
        TransTimeStamp='1999-08-09T12:58:40.000Z+9'>
    </TransId>
    <MsgId xml:lang='en' SoftwareID='Iotp wallet version 1.0'>
    </MsgId>
  </TransRefBlk>
  <IotpSignatures>
```

```
        <Signature>
            <Manifest>
                <Algorithm name='urn:nist-gov:sha1'
                                      type='digest' ID='P.3'>
                </Algorithm>
                <Algorithm name='urn:nist-gov:dsa'
                                      type='signature' ID='P.4'>
                    <Parameter type='AlgorithmRef'>P.5</Parameter>
                </Algorithm>
                <Algorithm name='urn:ibm-com:dom-hash'
                                      type='digest' ID='P.5'>
                    <Parameter type='AlgorithmRef'>P.3</Parameter>
                </Algorithm>
                <Digest DigestAlgorithmRef='P.6'>
                    <Locator href='P.1'/>
                    <Value>
                     xsqsfasDys2h44u4ehJDe54he5j4dJYTJ
                    </Value>
                </Digest>
                <OriginatorInfo
                    <IssuerAndSerialNumber
                     issuer='o=Iotp Ltd., c=US'
                     number='12345678987654'/>
                </OriginatorInfo>
                <RecipientInfo
                    SignatureAlgorithmRef='P.4'
                </RecipientInfo>
            </Manifest>
            <Value>
                9dj28fjakA9sked0Ks01k2d7a0kgmf9dk19lf63kkDSs0
            </Value>
        </Signature>
        <Certificate type='urn:X500:X509v3'>
            <IssuerAndSerialNumber
                issuer='o=GlobeSet Inc., c=US'
                number='123456789102356'/>
            <Value>
             xsqsfasDys2h44u4ehJDe54he5j4dJYTJ=
            </Value>
        </Certificate>
    </IotpSignatures>
    <PayExchBlk ID='P.1'>
        <PaySchemeData
            ID='P.2'
            PaymentRef='M.5'
            ContentSoftwareId='abcdefg'>
                <PackagedContent Name='FirstPiece'>
                    snroasdfnas934k
```

```
                    </PackagedContent>
            </PaySchemeData>
        </PayExchBlk>
    </IotpMessage>
```

7. Signature DTD

```
    <!--
    *****************************************************
    * IOTP SIGNATURES BLOCK DEFINITION                  *
    *****************************************************
    -->

    <!ELEMENT IotpSignatures (Signature+ ,Certificate*) >
    <!ATTLIST IotpSignatures
            ID          ID          #IMPLIED
    >

    <!--
    *****************************************************
    * IOTP SIGNATURE COMPONENT DEFINITION               *
    *****************************************************
    -->

    <!ELEMENT Signature (Manifest, Value+) >
    <!ATTLIST Signature
            ID          ID          #IMPLIED
    >

    <!ELEMENT Manifest
            (       Algorithm+,
                    Digest+,
                    Attribute*,
                    OriginatorInfo,
                    RecipientInfo+
            )
    >

    <!ATTLIST Manifest
            LocatorHRefBase         CDATA               #IMPLIED
    >

    <!ELEMENT Algorithm (Parameter*) >
    <!ATTLIST Algorithm
            ID                      ID                  #REQUIRED
            type                    (digest|signature)  #IMPLIED
            name                    NMTOKEN             #REQUIRED
    >
```

```
   <!ELEMENT Digest (Locator, Value) >
   <!ATTLIST Digest
         DigestAlgorithmRef    IDREF              #REQUIRED
   >

   <!ELEMENT Attribute ANY >
   <!ATTLIST Attribute
         type                  NMTOKEN            #REQUIRED
         critical          ( true | false )       #REQUIRED
   >

   <!ELEMENT OriginatorInfo ANY >
   <!ATTLIST OriginatorInfo
         OriginatorRef         NMTOKEN            #IMPLIED
   >

   <!ELEMENT RecipientInfo ANY >
   <!ATTLIST RecipientInfo
         SignatureAlgorithmRef  IDREF             #REQUIRED
         SignatureValueRef      IDREF             #IMPLIED
         SignatureCertRef       IDREF             #IMPLIED
         RecipientRefs          NMTOKENS          #IMPLIED
   >

   <!ELEMENT KeyIdentifier EMPTY>
   <!ATTLIST KeyIdentifier
         value                 CDATA              #REQUIRED
   >

   <!ELEMENT Parameter ANY >
   <!ATTLIST Parameter
         type                  CDATA              #REQUIRED
   >

   <!--
   ****************************************************
   * IOTP CERTIFICATE COMPONENT DEFINITION            *
   ****************************************************
   -->

   <!ELEMENT Certificate
     ( IssuerAndSerialNumber, ( Value | Locator ) )
   >

   <!ATTLIST Certificate
         ID                    ID                 #IMPLIED
         type                  NMTOKEN            #REQUIRED
   >
```

```
     <!ELEMENT IssuerAndSerialNumber EMPTY >
     <!ATTLIST IssuerAndSerialNumber
             issuer                      CDATA           #REQUIRED
             number                      CDATA           #REQUIRED
     >


     <!--
     ****************************************************
     * IOTP SHARED COMPONENT DEFINITION                 *
     ****************************************************
     -->
     <!ELEMENT Value ( #PCDATA ) >
     <!ATTLIST Value
             ID              ID          #IMPLIED
             encoding    (base64|none     'base64'
     >


     <!ELEMENT Locator EMPTY>
     <!ATTLIST Locator
             xml:link        CDATA       #FIXED          'simple'
             href            CDATA       #REQUIRED
     >
```

8. Security Considerations

   This entire document concerns the IOTP v1 protocol signature element
   which is used for authentication. See the Security Considerations
   section of [RFC 2801] "Internet Open Trading Protocol - IOTP, Version
   1.0".

References

   [DSA]   Federal Information Processing Standards Publication
                 FIPS PUB 186, "Digital Signature Standard(DSS)", 1994,
                 <http://csrc.nist.gov>

   [IEEE P1363] IEEE P1363, "Standard Specifications for Public-Key
                 Cryptography", Work in Progress, 1997,
                 <http://stdsbbs.ieee.org/>

   [PV]   Preneel, B. and P. van Oorschot, "Building fast MACs
                 from hash functions", Advances in Cryptology --
                 CRYPTO'95 Proceedings, Lecture Notes in Computer
                 Science, Springer-Verlag Vol.963, 1995, pp. 1-14.

   [RFC 1321]   Rivest, R., "The MD5 Message-Digest Algorithm", RFC
                 1321, April 1992.

   [RFC 2045]   Freed, N. and N. Borenstein, "Multipurpose Internet Mail
                 Extensions (MIME) Part One: Format of Internet Message
                 Bodies", RFC 2045, November 1996.

   [RFC 2046]   Freed N. and N. Borenstein, "Multipurpose Internet Mail
                 Extensions (MIME) Part Two: Media Types", RFC 2046,
                 November 1996.

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

   [RFC 2141]   Moats, R., "URN Syntax", RFC 2141, May 1997.

   [RFC 2253]   Wahl, W., Kille, S. and T. Howes, "Lightweight Directory
                 Access Protocol (v3): UTF-8 String Representation of
                 Distinguished Names", RFC 2253, December 1997.

   [RFC 2396]   Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform
                 Resource Identifiers (URI): Generic Syntax", RFC 2396,
                 August 1998.

   [RFC 2437]   Kaliski, B. and J. Staddon, "PKCS #1: RSA Cryptography
                 Specifications, Version 2.0", RFC 2437, October 1998.

   [RFC 2801]   Burdett, D., "Internet Open Trading Protocol - IOTP,
                 Version 1.0", RFC 2801, April 2000.

   [RFC 2803]   Maruyama, H., Tamura, K. and N. Uramot, "Digest Values
                 for DOM (DOMHASH)", RFC 2803, April 2000.

    [Schneier]   Bruce Schneier, "Applied Cryptography: Protocols,
                   Algorithms, and Source Code in C", 1996, John Wiley and
                   Sons

    [SHA1]   NIST FIPS PUB 180-1, "Secure Hash Standard," National
                   Institute of Standards and Technology, U.S. Department
                   of Commerce, April 1995.

    [X.509]   ITU-T Recommendation X.509 (1997 E), "Information
                   Technology - Open Systems Interconnection - The
                   Directory: Authentication Framework", June 1997.

    [X9.30]   ASC X9 Secretariat: American Bankers Association,
                   "American National Standard for Financial Services -
                   Public Key Cryptography Using Irreversible Algorithms
                   for the Financial Services Industry - Part 1: The
                   Digital Signature Algorithm(DSA)", 1995.

    [X9.62]   ASC X9 Secretariat: American Bankers
                   Association,"American National Standard for Financial
                   Services - Public Key Cryptography Using Irreversible
                   Algorithms for the Financial Services Industry - The
                   Elliptic Curve Digital Signature Algorithm (ECDSA)",
                   Work in Progress, 1997.

    [XLink]   Eve Maler, Steve DeRose, "XML Linking Language (XLink)",
                   <http://www.w3.org/TR/1998/WD-xlink-19980303>

    [XML]   Tim Bray, Jean Paoli, C. M. Sperber-McQueen, "Extensible
                   Markup Language (XML) 1.0",
                   <http://www.w3.org/TR/1998/REC-xml-19980210>

RFC 2802   Digital Signatures for IOTP   April 2000

Authors' Addresses

    The authors of this document are:

    Kent M. Davidson
    Differential, Inc.
    440 Clyde Ave.
    Mountain View, CA 94043 USA

    EMail: kent@differential.com

    Yoshiaki Kawatsura
    Hitachi, Ltd.
    890-12 Kashimada Saiwai Kawasaki,
    Kanagawa 2128567 Japan

    EMail: kawatura@bisd.hitachi.co.jp

Davidson & Kawatsura   Informational   [Page 28]

Full Copyright Statement

Acknowledgement