# XML Security Use Cases and Requirements

## W3C Working Draft 26 February 2009

**This version:**
http://www.w3.org/TR/2009/WD-xmlsec-reqs-20090226/
**Latest version:**
http://www.w3.org/TR/xmlsec-reqs/
**Editors:**
Frederick Hirsch, Nokia
Thomas Roessler, W3C

## Abstract

This Note summarizes scenarios, design decisions, and requirements for the XML Signature and Canonical XML specifications, to guide ongoing W3C work to revise these specifications.

## Status of this Document

*This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.*

*This is a First Public Working Draft of "XML Security Use Cases and Requirements."*

This document is expected to be further updated based on both Working Group input and public comments. The Working Group anticipates to eventually publish a stabilized version of this document as a W3C Working Group Note.

The use cases and requirements in this document are expected to help guide the XML Security Working Group's development of a version 2 of XML Signature. This Working Draft is published to solicit early community review.

This document was developed by the XML Security Working Group.

Please send comments about this document to public-xmlsec-comments@w3.org (with public archive).

Publication as a Working Draft does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. The group does not expect this document to become a W3C Recommendation. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

## Table of Contents

1 Introduction
2 Principles
3 Usage Requirements and Design Options
  3.1 Long Term Signatures
    3.1.1 General Requirements
    3.1.2 Proving that the signature was generated before a certain point of time.
    3.1.3 Proving that the signature was first verified before a certain point of time.
    3.1.4 Countering algorithm / keys weaknesses or validation material expiration
    3.1.5 XAdES and long term XML Signatures
  3.2 Web Services Security
    3.2.1 Assumptions
    3.2.2 Requirements
  3.3 Derived Keys
    3.3.1 Use Cases and Background
    3.3.2 Use Of Derived Keys in Existing WS-* Specifications
      3.3.2.1 Web Services Security: UsernameToken Profile Version 1.1
      3.3.2.2 WS-Trust Version 1.3:
      3.3.2.3 WS-SecurityPolicy 1.2:
      3.3.2.4 WS-SecureConversation 1.3:
    3.3.3 Solution Requirements
      3.3.3.1 Use in existing specifications
      3.3.3.2 No external dependencies
      3.3.3.3 Continued use of existing derivation methods
      3.3.3.4 Future-proof with regards to key lengths
      3.3.3.5 Referential flexibility
    3.3.4 Existing Specifications vs. Requirements
    3.3.5 Design Options
      3.3.5.1 Create a ds:DerivedKeyType type modeled after the xenc:EncryptedKeyType.
  3.4 Transforms
  3.5 Algorithm security and interoperability
    3.5.1 Fundamentals
    3.5.2 Requirements
      3.5.2.1 Address SHA security concerns, recognize RSA de-facto use.
      3.5.2.2 Revise guidance for DSAwithSHA1
      3.5.2.3 Add Suite B algorithm support
    3.5.3 Suite B Elliptic Curve Key Value Design (ECKeyValue)
      3.5.3.1 RFC 4050 issues in XML Signature context
      3.5.3.2 Proposed Solution to RFC 4050 issues in XML Signature context
4 Acknowledgments
5 References

## 1 Introduction

This use case and requirements document is intended to summarize use cases and requirements driving revisions to XML Signature 2nd Edition [XMLDSIG2nd] , XML Encryption [XMLENC] , and Canonical XML 1.1 [C14N11]. It is not intended to define all possible use cases for these Recommendations, but rather to provide rationale for decisions leading to XML Signature 1.1, XML Encryption 1.1 and XML Signature 2.0 and/or other specifications.

This document outlines general principles, elaborating on those expressed for the original XML Security work and then use cases, requirements and design options that impact the design of revisions to the XML Security specifications. It is a work in progress.

## 2 Principles

The following design principles will be used to guide further development of XML Security, including XML Signature, XML Encryption and Canonical XML. These principles are intended to encourage consistent design decisions, to provide insight into design rationale and to anchor discussions on requirements and design. This list includes items from the original requirements for XML Signature [XMLDSIG-REQS] as well as general principles from EXI [EXI]. Listed in alphabetical order:

**Backward compatible:**

Backward compatibility should not be broken unnecessarily. Versioning should be clearly considered. Consideration must be given, for example, for interoperability with the First and Second Editions of XML Signature [XMLDSIG2nd].

**Consistent with the Web Architecture:**

XML Security must be consistent with the Web Architecture [Webarch].

**Efficient:**

XML Security should enable efficient implementations, in order to remove barriers to adoption and use.

**Meet common requirements, enable extensibility:**

One of primary objectives of XML Signature is to support a wide variety of use cases requiring digital signatures, including situations requiring multiple signatures, counter-signatures, and signatures including multiple items to be included in a signature. Extensibility should be possible, but by default options should be constrained when the flexibility is not needed.

**Minimal:**

To reach the broadest set of applications, reduce the security threat footprint and improve efficiency, simple, elegant approaches are preferred to large, analytical or complex ones.

**Pragmatic:**

Recognize pragmatic issues, including recognizing that software might be implemented in layers, with a security layer independent of an application layer.

**Reuse Existing Open Standards**

Existing open standards should be reused where possible, as long as other principles can be met.

**Secure:**

XML Security should adhere to security best practices, and minimize the opportunities for threats based on XML Security mechanisms.

**XML Interoperable:**

XML Security must integrate well with existing XML technologies, be compatible with the XML Information Set [Infoset], in order to maintain interoperability with existing and prospective XML specifications.

**XML Signatures are First Class Objects:**

XML Signatures should themselves be self-describing first class XML objects [XMLDSIG-REQS]. This means that XML Signatures can be referenced via URI and used in other operations. For example, an XML Signature may be signed or encrypted, or referred to in a statement (such as an RDF statement).

## 3 Usage Requirements and Design Options

This section summarizes a number of scenarios in which the XML Signature and Canonical XML specifications are used, and identifies distinguishing properties of these scenarios.

Usage scenarios may fall into the following categories which may have different requirements:

- Document workflow

- Long Term Signatures
- Minimal signatures on XML or binary content
- Token signing and verification
- Web services

## 3.1 Long Term Signatures

Long term signatures are signatures whose contents allow an entity to ascertain, long time after the signature was first verified, that the signature was valid when produced.

The “long term” expression typically implies periods of several years (in certain European countries, invoices must be kept for up to five years, and this requirement is extended to electronic invoices electronically signed). These signatures must counter the effects of time passing. One of them, for instance, is the fact that after their first verification and after such a long period of time, certificates within the certificate path may have expired or some of them may even have been revoked.

The ascertaining entity may be, in some cases, different from the relying party that first verified the signature (an arbitrator in charge of solving disputes between signer and verifier, for instance).

### 3.1.1 General Requirements

There are general requirements associated with long term signatures:

1. There must be deployed secure mechanisms for proving that the signature was generated before a certain point in time.
2. There must be deployed secure mechanisms for proving that the signature was first verified by the corresponding relying party before a certain point in time.
3. There must be deployed mechanisms allowing the ascertaining entity to gain access to all the validation material that the relying party had used for verifying the signature.
4. There must be deployed mechanisms allowing to counter weaknesses discovered on cryptographic algorithms or keys used in the generation of the signature, or expiration of verification material.

Generally speaking each of the generic requirements in the list above may imply the incorporation of additional information within the XML Signature structure after the ds:SignatureValue has been generated.

### 3.1.2 Proving that the signature was generated before a certain point of time.

A long term signature must securely prove that it was actually generated before a certain point in time.

At present, two mechanisms have been identified for achieving such a requirement:

1. The inclusion in the signature of a trusted time-stamp, generated by a Trusted Time-Stamping Authority, immediately after the signature has been generated (signature time-stamp).
2. To deploy a process that generates artifacts that actually may prove in the future that the signature was created at that point of time (secure records of actions, for instance).

### 3.1.3 Proving that the signature was first verified before a certain point of time.

For resolving potential disputes between signer and verifier, a long term signature must also be able to securely prove that the relying party first verified it before a certain point in time.

For proving this, it must be securely proved that the verifier had gained access to all the verification material before that point of time. The requirements on the signature are, in consequence, as follows:

1. Mechanisms must be provided for collecting within the XML Signature, all the validation material used by the relying party in the first verification. This includes:
    a. Certificates within the certificate path of the signing certificate.
    b. Material reporting the status of each certificate within the certificate path of the signing certificate (CRLs or OCSP responses).
    c. Certificates within the certificate path of the certificate used by the Trusted Time-Stamping Authority that generated the signature time-stamp.
    d. Material reporting the status of each certificate within the certificate path aforementioned in c.
2. A trusted time-stamp generated by a Trusted Time-Stamping Authority, covering both the signature generated and the verification material listed before. This time-stamp will actually prove that the relying party had gained access to the validation material and in consequence, had the capability for verifying the signature, before the time indicated in the time-stamp.

#### 3.1.4 Countering algorithm / keys weaknesses or validation material expiration

Once the validation material has been collected and time-stamped, the action of time may bring two different types of consequences:

1. Some of the algorithms (or keys) used for generating the signature or any of the aforementioned time-stamps may be broken.
2. Some of the certificates in the certification path of the time-stamp covering this validation material may expire.

A long-term signature must counter any of the two consequences in the list.

A countering mechanism consists in regularly generating trusted time-stamps covering the signature, all the previously issued time-stamps and all the validation material (archive time-stamps).

If some algorithm / key has been broken, this time-stamp may use different algorithm / keying material.

If some of the certificates within the certificate path of the latest time-stamp have expired, the new one will have a new certificate within its corresponding certificate path.

A form of long term signatures may be obtained by allowing XML Signatures to incorporate such archive time-stamps.

#### 3.1.5 XAdES and long term XML Signatures

XAdES, "XML Advanced Electronic Signatures" [XAdES], specifies XML Schema types suitable for containing all the different types of data mentioned in here, namely, time-stamp tokens, validation data, etc.

XAdES also standardizes ways for adding all this material to regular XML Signatures, increasing interoperability among applications.

## 3.2 Web Services Security

#### 3.2.1 Assumptions

1. Message content will be provided and processed by multiple software components acting autonomously. The XML will make use of multiple namespaces, potentially with duplicate element names.
2. Messages may pass through multiple intermediary nodes which may add, subtract or alter content in either the SOAP header or body.

#### 3.2.2 Requirements

1. Generally the ability to provide ephemeral authentication, integrity protection and confidentiality of message content including attachments, using a variety of technologies. In some cases, messages with signatures may be stored for purposes of non-repudiation.
2. Any or all of messages may be signed and/or encrypted zero or more times in any order. Signatures and encryptions may overlap. A receiver must be able to properly verify signatures and decrypt data in the proper order (assuming access to the necessary secrets or trust points) based on nothing but the message.
3. It must be possible to determine whether the correct portions of the message have been signed and encrypted with the correct keys according to policy.
4. To the extent possible allowed by the ordering of data and cryptographic operations it should be possible for a sender or a receiver to perform processing in a single pass over the message.

## 3.3 Derived Keys

### 3.3.1 Use Cases and Background

Several open specifications make use of derived keys, e.g. RSA Laboratories' PKCS #5 v2.0 [PKCS#5] and OASIS' WS-SecureConversation Version 1.3 [WS-SecureConversation13]. These derived keys are used for a variety of purposes including encryption and message authentication, and the purpose of key derivation itself is typically a combination of a desire to expand a given, but limited, set of key material and prudent security practices of limiting use (exposure) of such key material.

Contrary to the situation in the ASN.1-based world (e.g. S/MIME [S/MIME]), there is currently a lack of general support in the core XML Security specifications, XML Signature and XML Encryption, for derived keys. Amendment 1 of the aforementioned PKCS #5 v2.0 Amendment 1 [PKCS#5] adds support for derived keys only in the context of password-based cryptography. Other XML-based open specifications have similar limitations (see below). This means that an originator of an XML document or message cannot generally make use of key derivation in a standardized manner when performing cryptographic operations on that document.

### 3.3.2 Use Of Derived Keys in Existing WS-* Specifications

#### *3.3.2.1 Web Services Security: UsernameToken Profile Version 1.1*

This specification [WSS-Username11] describes a key derivation technique for passwords using salt and iteration count (PKCS #5 PBKDF1). It does not allow use of PBKDF2, which is the recommended method to derive keys from passwords in PKCS #5 v2.0. Initial key material cannot be referenced other than with wsu:Id. The key length will always be 160 bits.

#### *3.3.2.2 WS-Trust Version 1.3:*

Ws-Trust Version 1.3 [WS-Trust13] describes key derivation through a combination of entropies from both parties. The key is never sent on the wire. The key is never referenced directly (but further key material is derived from it). WS-Trust provides one specific method to derive keys which builds on the P_hash (P_SHA-1) function from TLS.

#### *3.3.2.3 WS-SecurityPolicy 1.2:*

WS-SecurityPolicy Version 1.2 [WS-SecurityPolicy12] really only specifies whether derived keys shall be used or not but may also specify the algorithm to derive keys. The specification also identifies when derived key tokens shall appear in message headers (header layout). WS-SecurityPolicy relies on WS-SecureConversation for the definition of derived keys, key derivation methods and derived key token format.

#### *3.3.2.4 WS-SecureConversation 1.3:*

This specification [WS-SecureConversation13] defines the wsc:DerivedKeyTokenType token type. The derived key token can be used to derive keys from any other token that contains keys. The key derivation algorithm specified builds on the P_hash (P_SHA-1) function from TLS, just as the algorithm in the Web Service Security UsernameToken Profile document. Citing from the specification: "The `<wsc:DerivedKeyToken>` element is used to indicate that the key for a specific reference is generated from the function. This is so that explicit security tokens, secrets, or key material need not be exchanged as often." (This latter seems dubious since the DerivedKeyToken still needs to be exchanged.) Further: "Basically, a signature or encryption references a `<wsc:DerivedKeyToken>` in the `<wsse:Security>` header that, in turn, references the `<wsc:SecurityContextToken>`." The derived key token does not support references using key identifiers or key names. All references MUST use an ID (to a wsu:Id attribute) or a URI reference to the `<wsc:Identifier>` element in the Security Context Token.

### 3.3.3 Solution Requirements

#### *3.3.3.1 Use in existing specifications*

A derived key type shall be possible to use in those situations where existing specifications make use of ad-hoc derived keys or needs a derived key type

The motivation for this requirement is that any XML Security definition shall be generic enough that there shall be no need to continue with "point" solutions for derived keys; i.e. it shall cover existing and foreseeable uses.

#### *3.3.3.2 No external dependencies*

A derived key type shall enable the simple use of derived keys with XML Signature or XML Encryption -using applications, and shall not require import of non-W3C developed specifications with complex security tokens.

The motivation for this is that basic use of XML Signature or XML Encryption should not require use of externally defined security tokens or other security specification elements.

#### *3.3.3.3 Continued use of existing derivation methods*

An XML Security derived key type shall allow for existing methods to derive keys; i.e. it shall be possible to use already specified key derivation methods with the new derived key type.

This requirement is based on the assumptions that implementations may want to continue with already chosen key derivation schemes.

#### *3.3.3.4 Future-proof with regards to key lengths*

A derived key type shall allow for arbitrary derived key lengths.

#### *3.3.3.5 Referential flexibility*

A derived key type shall allow for referencing using any referencing method in use today for other key types used in XMLDsig or XMLEnc.

A derived key type shall allow for forward referencing with reference lists as recommended by WS-I BSP [WS-I-BSP10].

### 3.3.4 Existing Specifications vs. Requirements

Evaluating the existing specifications against the requirements gives the following result:

UsernameToken Profile:

- R1: Not met (method specified in UsernameToken profile is ad-hoc for UsernameToken specifically)

- R2: Not met (method requires use of UsernameToken profile)
- R3: Not met (UsernameToken profile mandates use of specified mechanism)
- R4: Not met (Only accept length of 160 bits)
- R5: Not met (No referencing with KeyName or KeyIdentifier and no `<referenceList>` element)

WS-Trust:

- R1: N/A (WS-Trust does not define a derived key type per se; only a method to derive keys)
- R2: N/A
- R3: Meets (Through use of URI to identify method and extensibility)
- R4: Meets
- R5: Meets (Choice of STS on how to identify key)

WS-SecurityPolicy:

- R1: N/A (WS-SecurityPolicy does not define a derived key type)
- R2: N/A
- R3: Meets (Through the use of URIs to identify key derivation methods and schema extensibility)
- R4: Meets
- R5: N/A

WS-SecureConversation:

- R1: Meets
- R2: Does not meet.
- R3: Meets (may use the `<Properties>` element to carry parameters for other key derivation methods.
- R4: Meets
- R5: Does not meet as referencing can only be done to a `<wsse:SecurityTokenReference>`

### 3.3.5 Design Options

#### *3.3.5.1 Create a ds:DerivedKeyType type modeled after the xenc:EncryptedKeyType.*

In this design option, the new DerivedKeyType is modeled after the xenc:EncryptedKeyType. A *possible* outline of such a type could be:

**Example: Outline of possible DerivedKeyType schema definition**

```
<element name="DerivedKey" type="xmlsec:DerivedKeyType"/>
<complexType name="DerivedKeyType">
   <sequence>
     <element name="KeyDerivationMethod" type="xmlsec:KeyDerivationMethodType" minOccurs="0"/>
     <element ref="xenc:ReferenceList" minOccurs="0"/>
     <element name="CarriedKeyName" type="string" minOccurs="0"/>
   </sequence>
   <attribute name="Id" type="ID" use="optional"/>
   <attribute name="Type" type="anyURI" use="optional"/>
</complexType>

<complexType name="KeyDerivationMethodType">
```

```
    <sequence>
      <any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="Algorithm" type="anyURI" use="required"/>
 </complexType>
```

The proposal immediately meets requirements R2, R3 (any key derivation method may be used, including the ones specified, e.g., in WS-SecureConversation), R4 and R5. For R1 we have:

Username Token Profile: As the UsernameToken Profile requires use of an existing procedure to derive keys, the proposal cannot meet formally meet requirement R1. However, since the UsernameTokenType is extensible, syntactically the requirement can be met since a `<ds:DerivedKey>` element could be placed in lieu of the current `<salt>` and `<iteration>` elements.

WS-Trust: Use of derived keys in WS-Trust is _implicit_, since the derived key is never sent. The derived keys may be referenced by any available means in issued tokens and the requestor is only required to identify particular key derivation methods. Since URIs are used for this (the `<wst:ComputedKey>` element), any other key derivation method with a well-known URI may be used. Specifically, one can also envision an STS returning a proof token containing a `<DerivedKey>` element when there already is a shared key between the STS and a token requestor. And so, R1 is met.

WS-SecurityPolicy: Not affected by a new key type. R1 is met.

WS-SecureConversation: Use of derived keys in WS-SecureConversation is typically based on the establishment of a session context, from which specific keys are derived. The proposed `<xmlsec:DerivedKeyType>` type may be used in a similar fashion, although the interactive nature of WS-SecureConversation (exchange of Nonces, Labels) may still favor use of the existing DerivedKeyToken in this context. But as a counterexample, a party that wishes to send data authenticated with a key derived from a key established in the session, may do so using the `<xmlsec:DerivedKey>` element in the `<ds:KeyInfo>` element, and the element may refer to a SecurityContextToken that identifies the base key. This would, it seems, eliminate an absolute need for a `<wsc:DerivedKeyToken>` (and should be similar in nature as the "Implied Derived Key" option in WS-SecureConversation). Also, the `<wsc:DerivedKeyToken>` implies use of a particular key derivation algorithm (the `<Label>` and `<Nonce>` elements) although it does not require them.

In summary, WS-Trust and WS-SecurityPolicy are not directly affected by this proposal. UsernameToken profile could use the proposal if the (artificial) requirement to only use the key derivation method specified in the UsernameToken Profile document was relaxed. WS-SecureConversation comes close in establishing an alternative but the specification defines a token primarily for use in interactive sessions based on a security context and which is designed for a particular key derivation method. It also seems strange to require use of such a token in more basic XMLDsig or XMLEnc situations. Finally, the proposal seems to be able to replace the DerivedKeyToken currently used in WS-SecureConversation.

## 3.4 Transforms

Usage scenarios, requirements, issues and design related to XML Signature transform processing are discussed in a separate document [TransformSimplification]. Those requirements should be considered in conjunction with those in this document.

## 3.5 Algorithm security and interoperability

### 3.5.1 Fundamentals

XML Signature specifies algorithm identifiers and implementation requirements for algorithms related to various aspects of signature processing, including digest and signature algorithms. The algorithms listed in XML Signature, Second Edition date from the original XML Signature Recommendation, published in 2002. Since that time there have been new algorithms introduced to address security risks associated with earlier algorithms (e.g. SHA-256 versus SHA-1), changes in patent status related to algorithms (e.g. RSA signing no longer has licensing requirements), and additional algorithms introduced to meet additional requirements (Suite B algorithms [SuiteB]).

In order to meet the principle of "Secure" and "Pragmatic", new algorithm requirements should be met.

### 3.5.2 Requirements

*3.5.2.1 Address SHA security concerns, recognize RSA de-facto use.*

In order to address concerns related to potential risks associated with SHA-1, the following algorithm requirements that update the SHA algorithm should be met in XML Signature:

- Digest:

  SHA256 be required.

  SHA384 and SHA512 optional.

- Mac:

  HMAC-SHA256 recommended.

  HMAC-SHA384 and HMAC-SHA512 optional.

- Signature:

  RSAwithSHA256 required.

  RSAwithSHA384, RSAwithSHA512 optional.

*3.5.2.2 Revise guidance for DSAwithSHA1*

In order to discourage the use of DSAwithSHA1 but to continue to enable interoperability, the following algorithm changes are requirements;

- Signature:

  Continue to require DSAwithSHA1 for signature verification, but change DSAwithSHA1 to optional (from required) for signature generation.

*3.5.2.3 Add Suite B algorithm support*

In order to enable use of XML Signature technology in interoperable US government applications that require Suite B, and to enable long term security for commercial companies, elliptic curve algorithms are to be added to XML Signature. As new hardware is developed and new algorithms to break cryto systems are found, the stronger algorithms offered by elliptic curve enable longer term security.

The additional algorithm requirements are as follows:

- Signature:

  Require ECDSAwithSHA256.

  ECDSAwithSHA1, ECDSAwithSHA384, ECDSAwithSHA512 optional.

- Define ECKeyValue element to enable interoperable exchange of EC public key values in XML Signature context.

- Provide profile guidance for use of RFC 4050 [RFC4050] when it continues to be used in XML Signature context but indicate preference for mechanism defined in XML Signature.

The last two requirements are discussed in more detail in the following design section.

### 3.5.3 Suite B Elliptic Curve Key Value Design (ECKeyValue)

### *3.5.3.1 RFC 4050 issues in XML Signature context*

RFC 4050 is an informational RFC that defines a method of representing ECDSA public keys and ECC curve parameters for use with XML Signature, but it has some issues related to XML Signature:

- The RFC 4050 definition of an ECDSAKeyValue is larger than necessary.

  An ECDSAKeyValue is defined by the type ECPointType, which has subelements X and Y. X and Y are defined as FieldParamsType which is an abstract type. Separate derived types are defined for prime fields, trinomial base fields, pentanomial base fields, and odd characteristic extension fields. In order to validate against the 4050 schema, one must include the type attribute from the XML schema instance namespace. This is not a significant problem but it does make the public key larger than necessary.

- ECPointType definition is inconsistent with ANSI X9.62 and RFC 3279.

  ECPointType is reused in the definition of the ExplicitParamsType to describe the base point of a curve. The field parameters are already included in the FieldParams element. The use of the FieldParamsType in the ECPointType definition appears to be a mistake in 4050. If you look at the ASN.1 definition for ECC public keys in RFC 3279 [RFC3279], ECPoint simply references the Point to Octet String conversion function in ANSI X9.62 (section A.5.6 in the 2005 version, section 4.3.6 in the 1998 version). The conversion functions in X9.62 are not ASN.1 specific and it appears they would be implemented as part of any ECC crypto library. It appears that RFC 4050 tried to avoid using any of the conversion functions in X9.62 but somehow mixed up the definitions between a field type and a field element.

- Limitation of the decimal type in XSD

  RFC 4050 defines X and Y (at least for prime and odd characteristic extension fields) as xs:nonNegativeInteger which derives from the xs:decimal primitive type. However, XSD requires implementations to support only a maximum of 18 digits (see section 3.2.3 in [XSD]). It is possible to create an example requiring 77 and 78 digits for X and Y respectively. This means that there is no guarantee that an RFC 4050 compliant ECDSAKeyValue element will actually validate against the RFC 4050 schema.

- Collision between the RFC 4050 DTD and the XMLDSIG DTD

  Merging the RFC 4050 DTD into the XMLDSIG DTD is a problem due to conflicting DTD definitions. In ECDSAKeyValue, Y is defined as follows:

  **Example: Definition of Y in ECDSAKeyValue**

  ```
  <!ELEMENT Y EMPTY>
  <!ATTLIST Y Value CDATA #REQUIRED>
  ```

  However, DSAKeyValue defines Y as follows:

  **Example: Definition of Y in DSAKeyValue**

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

  ECDSAKeyValue also contains identical definition for elements SEED and P as DSAKeyValue.

  It does not seem possible to scope the definition of Y under a specific element in DTD.

### *3.5.3.2 Proposed Solution to RFC 4050 issues in XML Signature context*

Because of these issues, the proposed solution is for XML Signature 1.1 to define a new ECPublicKey element in the ds namespace rather than attempt to reuse the RFC 4050 ECDSAPublicKey elements. This new element will be based on the ASN.1 definition ANSI X9.62 and RFC 3279. Changing the name of the element to ECPublicKey means it can be also used in XML Encryption to support ECDH.

To maximize interoperability with existing RFC 4050 implementations, we should also put a note in 1.1 to recommend implementations to support a profile of RFC 4050. The profile will support only named prime curves.

## 4 Acknowledgments

@@ TBD @@

## 5 References

**BradHill**
*Complexity as the Enemy of Security: Position Paper for W3C Workshop on Next Steps for XML Signature and XML Encryption*, Brad Hill, 25-26 September 2007, http://www.w3.org/2007/xmlsec/ws/papers/04-hill-isecpartners/

**C14N-REQS**
*XML Canonicalization Requirements*, James Tauber, Joel Nava. W3C Note, 5 June 1999, http://www.w3.org/TR/1999/NOTE-xml-canonical-req-19990605.

**C14N11**
*Canonical XML 1.1*, John Boyer, Glenn Marcy. W3C Recommendation 2 May 2008, http://www.w3.org/TR/2008/REC-xml-c14n11-20080502/.

**EXI**
*Efficient XML Interchange (EXI) Format 1.0*, W3C Working Draft 28 July 2008, J. Schneider, T. Kamiya http://www.w3.org/TR/2008/WD-exi-20080728/

**Gajek**
*Towards a Semantic of XML Signature: Position Paper for W3C Workshop on Next Steps for XML Signature and XML Encryption*, Sebastian Gajek, Lijun Liao, and Jörg Schwenk, 25-26 September 2007, http://www.w3.org/2007/xmlsec/ws/papers/07-gajek-rub/

**Infoset**
*XML Information Set (Second Edition)*, W3C Recommendation 4 February 2004. J. Cowan, R. Tobin http://www.w3.org/TR/2008/WD-exi-20080728/#XMLInfoset

**McIntoshAustel**
*XML signature element wrapping attacks and countermeasures.*M. McIntosh and P. Austel. In Workshop on Secure Web Services, 2005.

**PKCS#5**
*PKCS #5 v2.0: Password-Based Cryptography Standard*, RSA Laboratories, March 25, 1999. http://www.rsa.com/rsalabs/node.asp?id=2127, This page also references related amendments.

**RFC3279**
*Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, W. Polk, R. Housley, L. Bassham. IETF RFC 3279. April 2002. http://www.ietf.org/rfc/rfc3279.txt

**RFC4050**
*Using the Elliptic Curve Signature Algorithm (ECDSA) for XML Digital Signatures*, S. Blake-Wilson, G. Karlinger, T. Kobayashi, Y. Wang. IETF RFC 4050, April 2005. http://www.ietf.org/rfc/rfc4050.txt

**S/MIME**
*S/MIME Version 3 Message Specification*, RFC 2633, June 1999. http://www.ietf.org/rfc/rfc2633.txt?number=2633

**SuiteB**
*Fact Sheet NSA Suite B Cryptography*, http://www.nsa.gov/ia/industry/crypto_suite_b.cfm

**Thompson**
*Radical proposal for Vnext of XML Signature*, Henry Thompson. Position paper, 26 September 2007, http://www.w3.org/2007/xmlsec/ws/papers/20-thompson/.

**TransformSimplification**
*XML Signature Transform Simplification: Requirements and Design*, Editors Draft, http://www.w3.org/2008/xmlsec/Drafts/transform-note/Overview.html.

**WS-I-BSP10**
*Basic Security Profile Version 1.0*,Final Material, 2007-03-30. http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html

**WS-SecureConversation13**
*WS-SecureConversation 1.3*,OASIS Standard, 1 March 2007.

http://www.oasis-open.org/specs/index.php#wssecconv1.3

**WS-SecurityPolicy12**
*WS-SecurityPolicy 1.2*, OASIS Standard, 1 July 2007. http://www.oasis-open.org/specs/index.php#wssecpolv1.2

**WS-Trust13**
*WS-Trust 1.3*,OASIS Standard, 19 March 2007. http://www.oasis-open.org/specs/index.php#wstrustv1.3

**WSS**
*Web Services Security v1.1*, OASIS Standard, February 2006. http://www.oasis-open.org/specs/index.php#wssv1.1

**WSS-Username11**
*Web Services Security UsernameToken Profile 1.1*, OASIS Standard Specification, 1 February 2006. http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf

**Webarch**
*Architecture of the World Wide Web, Volume One*, W3C Recommendation 15 December 2004, I. Jacobs, N. Walsh. http://www.w3.org/TR/webarch/

**XAdES**
*ETS TS 101 903: "XML Advanced Electronic Signatures (XAdES)" v1.3.2 March 2006.*, http://webapp.etsi.org/workprogram/Report_WorkItem.asp?WKI_ID=21353

**XMLDSIG**
*XML-Signature Syntax and Processing*, D. Eastlake, J. R., D. Solo, M. Bartel, J. Boyer , B. Fox , E. Simon. W3C Recommendation, 12 February 2002, http://www.w3.org/TR/xmldsig-core/.

**XMLDSIG-REQS**
*XML-Signature Requirements*, Joseph Reagle. W3C Working Draft, 14 October 1999, http://www.w3.org/TR/xmldsig-requirements.

**XMLDSIG2nd**
*XML Signature Syntax and Processing (Second Edition)*, W3C Recommendation 10 June 2008 http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/

**XMLENC**
*XML Encryption Syntax and Processing* , D. Eastlake, J. Reagle, W3C Recommendation 10 December 2002, http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/.

**XMLSecNextSteps**
*Workshop Report W3C Workshop on Next Steps for XML Signature and XML Encryption*, W3C, 25-26 September 2007, http://www.w3.org/2007/xmlsec/ws/report.html

**XSD**
*XML Schema Part 2: Datatypes Second Edition*, Biron P, Malhotra A, W3C Recommendation 28 October 2004. http://www.w3.org/TR/xmlschema-2