Intelligence Community Technical Specification

Web Service Security Guidance for Use of XML Signature and XML Encryption

Version 1

09 May 2014

Table of Contents

List of Figures

List of Tables

Chapter 1 - Introduction

1.1 - Purpose

The purpose of this document is to provide guidance to solutions architects, integrators and developers on how to minimize the risks and the vulnerabilities with the use of XML Signature and XML Encryption.

The guidance provided by this document is at a high level, intended to provide an understanding of risks associated with the vulnerabilities of using XML Signature and XML Encryption. While this document does not provide low-level details needed for implementation, it points to lower-level specifications and standards for that necessary details, and it should be sufficient to act as a consistent basis upon which solutions architects, integrators and developers can design and implement specific security solutions.

1.2 - Scope

This information guidance document applies to solutions using World Wide Web Consortium (W3C) XML Signature[30] and XML Encryption[29] for XML message in transit though HTTP - based web service (SOAP [25] and REST [20]). The W3C XML Security Working Group develops updates to the core XML Security specifications, which include the W3C recommendations for XML Encryption, XML Signature and XML Signature Properties. The group also publishes working group notes that provide best practices guide, use cases, requirements and test cases for the specifications. For example, the technical note *XML Signature Best Practices* [31] collects best practices for implementers and users of the XML Signature specification.

This information guidance document focuses on the vulnerabilities of XML signature and XML encryption and how to minimize the risks. General guidelines are provided as well as guidelines for specific data standards and web service implementations. The document does not define new standard. It harmonizes existing work from open standards organizations and related DOD / IC specifications to address the interoperability and security needs of various use cases.

The following topics, while related with XML security, are out of the scope of the documents:

- Non- W3C solutions, e.g., Cryptographic Message Syntax (CMS) based signature and encryption,
- Guidelines for non-vulnerability issues, e.g., key management and performance optimization,
- Use cases using non- HTTP services, e.g., SFTP , SMTP or enterprise messaging,
- Advertisement and exchange of security policy.

1.3 - Background

The IC Chief Information Officer (IC CIO) is leading the IC's enterprise transformation to a flexible, scalable and interoperable architecture for use within and across the IC's environments. Intelligence Community Directive (ICD) 500, *Director of National Intelligence Chief Information Officer*, grants the IC CIO the authority and responsibility to:

- Develop an IC Enterprise Architecture (IC EA)
- Lead the IC's identification, development, and management of IC enterprise standards
- Incorporate technically sound, de-conflicted, interoperable enterprise standards into the IC EA
- Certify IC elements adhere to the architecture and standards.

In the area of enterprise standardization, the IC CIO is called upon to establish common IT standards, protocols, and interfaces; to establish uniform information security standards; and to ensure information technology infrastructure, enterprise architecture, systems, standards, protocols, and interfaces, support the overall information sharing strategies and policies of the IC as established in relevant law, policy, and directives.

1.4 - Enterprise Need

The IC CIO funds and oversees a number of critical enabling projects, including the IC Information Technology Enterprise (IC ITE) . The IC ITE makes extensive use of web services and distributed processing, yet each individual program providing services therein requires explicit guidance on building secure, interoperable web services.

This information guidance document provides guidance for the development of secure and interoperable web services security solutions in support of ICD 500[7], ICD 501, Intelligence Community Standard (ICS) 500-20, ICS 500-21, ICS 500-27 , Intelligence Community Program Guidance 500.1 (ICPG 500.1)[8] and ICPG 500.2.[9]

Enterprise needs and requirements for this specification can be found in the following Office of the Director of National Intelligence (ODNI) policies and implementation guidance.

- IC Information Technology Enterprise (IC ITE)
  - Intelligence Community Information Technology Enterprise (IC ITE) Increment 1 Implementation Plan[4]
- 500 Series:
  - Intelligence Community Directive (ICD) 500, Director Of National Intelligence Chief Information Officer[7]

1.5 - Audience and Applicability

The intended audience of this information guidance document is project managers, software architects, network architects, and developers who develop and integrate with web services. This document provides guidance in areas that will be important in satisfying security requirements and information security goals in a secure and interoperable manner.

The applicability of this information guidance document is defined in the IC Enterprise Standards Baseline (IC ESB) . Additional applicability and guidance may be defined in separate IC policies, as necessary.

ICS 500-20, Intelligence Community Enterprise Standards Compliance,[11] defines the IC ESB and its applicability to IC Elements. The IC ESB defines the compliance requirements

associated with each version of a technical specification. Each version will be individually registered in the IC ESB . The IC ESB defines the location(s) of the relevant artifacts, prescriptive status, and validity period, all of which characterize the version and its utility.

1.6 - Conventions

Certain technical and presentation conventions were used in the creation of this document to ensure readability and understanding.

1.6.1 - Language

The keywords “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” in this technical specification are to be interpreted as described in the IETF RFC 2119.[12] These implementation indicator keywords are thus capitalized when used to unambiguously specify requirements over protocol and application features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

1.6.2 - Typography

Certain typography is used throughout the body of this document to convey certain meanings, in particular:

- *Italics* – A title of a referenced work or a specialized or emphasized term
- Underscore – An abstract data element
- **Bold** – An XML element or attribute

1.6.3 - XML Namespaces

Namespaces referenced in this document and the prefixes used to represent them are listed in the following table. The namespace prefix of any XML Qualified Name used in any example in this document should be interpreted using the information below.

Table 1 - XML Namespaces

Prefix	URI	Description
atom	http://www.w3.org/2005/Atom	Atom Syndication Format
ds	http://www.w3.org/2000/09/xmldsig#	W3C XML Signature
saml	urn:oasis:names:tc:SAML:2.0:assertion	SAML 2.0
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd	WS -Security
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	WS -Security utility

Prefix	URI	Description
xenc	http://www.w3.org/2001/04/xmlenc#	W3C XML encryption 1.0
xenc11	http://www.w3.org/2009/xmlenc11#	W3C XML encryption 1.1

1.7 - Dependencies

This information guidance document refers to the additional documentation listed in Table 2 . The documents and standards listed below are referenced throughout this document.

Table 2 - Normative Dependencies

Name
Intelligence Community High Level Guidance for Web Service Security[6]
OASIS Security Assertion Markup Language (SAML) , version 2.0[22]
OASIS Binding for the Security Assertion Markup Language (SAML) v 2.0[SAML 2.0 binding]
OASIS Profiles for the Security Assertion Markup Language (SAML) v 2.0[23]
OASIS Web Services Security: SOAP Message Security Version 1.1.1[26]
OASIS Extensible Access Control Markup Language (XACML) v. 3.0[27]
ANSI / NIST ITL 2011 Type 98 Best Practice Implementation Guidance For the Assurance of Biometric Data Integrity, Authenticity and Auditable Chain[16]
W3C XML Encryption (XML Encryption Syntax and Processing), Version 1.1[29]
W3C XML Signature (XML Signature Syntax and Processing) Version 1.1[30]
RFC4287, December 2005, The Atom Syndication Format[13]

Table 3 - Informative Dependencies

Name
NIST Special Publication (SP) 800-38D: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC [14]
NIST Special Publication (SP) 800-57: Recommendation for Key Management – Part 1: General[15]
NIST FIPS PUB 140-2 Security Requirements for Cryptographic Modules[2]
NIST FIPS PUB 186-4 Digital Signature Standard (DSS) [3]
NSA Vol2 Cryptographic Binding CONOPS [17]
NSA Guidelines for Implementation of REST [18]
NSA REST White Paper[19]
IC Trusted Data Format (IC-TDF.XML) specification[5]
OASIS SAML 2.0 Technical Overview[24]
W3C XML Signature Best Practices, W3C Working Group Note[31]

1.8 - Conformance

For an implementation to conform to this specification, it MUST adhere to all normative aspects of the specification. For the purposes of this document, normative and informative are defined as:

- *Normative*: considered to be prescriptive and necessary to conform to the standard.
- *Informative*: serving to instruct, enlighten or inform.

Additional guidance that is either classified or having handling controls can be found in separate annexes, distributed to the appropriate networks and environments, as necessary. Systems and services operating in those environments MUST consult the appropriate annexes.

1.9 - Definitions

The following terms listed in this section are used throughout this information guidance document to provide clarity and consistency.

Table 4 - Definitions

Term	Definition
Assertion	Used to represent a claim that is propagated to a service provider for the purpose of informing an access control decision. (Source: IC-WSS-HLG.XML)
Authentication	The process of verifying the identity or other attributes claimed by or assumed of an entity (user, process, or device). (Source: IC-WSS-HLG.XML, CNNSI 4009)
Authorization	The assessment of permissions granted to and restrictions imposed on a subject that establishes whether a subject may carry out an action. (Source: IC-WSS-HLG.XML)
Confidentiality	The property that information is not disclosed to system entities (users, processes, devices) unless they have been authorized to access the information. (Source: IC-WSS-HLG.XML, CNSSI -4009)
Availability	Ensuring timely and reliable access to and use of information, and the property of being accessible and useable upon demand by an authorized entity. (Source: IC-WSS-HLG.XML, NIST SP 800-53, CNSSI -4009)
Integrity	The property whereby an entity has not been modified in an unauthorized manner (Source: IC-WSS-HLG.XML, CNSSI -4009)
Non-repudiation	Assurance that the sender of information is provided with proof of delivery and the recipient is provided with proof of the sender’s identity, so neither can later deny having processed the information. (Source: IC-WSS-HLG.XML, CNSSI -4009).
Cryptographic binding	Methodology for providing integrity and authenticity to data and data relationships using well-known cryptographic techniques. (Source: NSA CryptoBinding CONOPS)

Term	Definition
Security life of data	The time period during which the security of the data needs to be protected (e.g., its confidentiality, integrity or availability). (SOURCE: NIST SP800-57)
Security strength (also “bits of security”)	A number associated with the amount of work (that is, the number of operations) that is required to break a cryptographic algorithm or system. (SOURCE: NIST SP800-57)

Chapter 2 - Overview

2.1 - XML Signature and Encryption

The W3C has recommended and published two XML standards that improve Web Service Security:

- XML Signature - specifies XML syntax and processing rules for creating and representing digital signatures,
- XML Encryption - specifies a process for encrypting data and representing the result in XML.

XML Signatures can be applied to any digital content (data object), including XML. XML Signature is a method of associating a key with referenced data (octets); it does not normatively specify how keys are associated with persons or institutions, nor the meaning of the data being referenced and signed.

Encrypted data may be arbitrary data (including an XML document), an XML element, or XML element content. The result of encrypting data is an XML Encryption element (EncryptedData) that contains (via one of its children's content) or identifies (via a URI reference) the cipher data. When encrypting an XML element or element content the EncryptedData element replaces the element or content (respectively) in the encrypted version of the XML document. When encrypting arbitrary data (including entire XML documents), the EncryptedData element may become the root of a new XML document or become a child element in an application-chosen XML document. The following sections describe these two W3C XML standard concepts in detail.

2.1.1 - Introduction to XML Signature

The W3C Recommendation: *XML -Signature Syntax and Processing* (abbreviated in this document as XMLDSig)[30] specifies XML digital signature processing rules, syntax and algorithms. XML signatures provide integrity, message authentication and signer authentication services for XML data. XMLDSig supports multiple signatures in the same XML document and it permits different entities to sign distinct portions of a single document. The signature information is an XML fragment with a **Signature** element as the root. A **SignatureValue** element contains the actual value of the digital signature. A **SignedInfo** child element includes information on the signature creation process, e.g., the canonicalization algorithm, a signature algorithm, and one or more **Reference**s. Finally, an optional **KeyInfo** child element of the signature enables the recipient(s) to obtain the key needed to validate the signature.

A **Reference** element under **SignedInfo** specifies a digest algorithm and digest value, and optionally an identifier of the object being signed, the type of the object, and/or a list of transforms to be applied prior to digesting. Examples of transforms include but are not limited to base64 decoding, canonicalization, XPath filtering, XSLT, and application-specific transform algorithms.

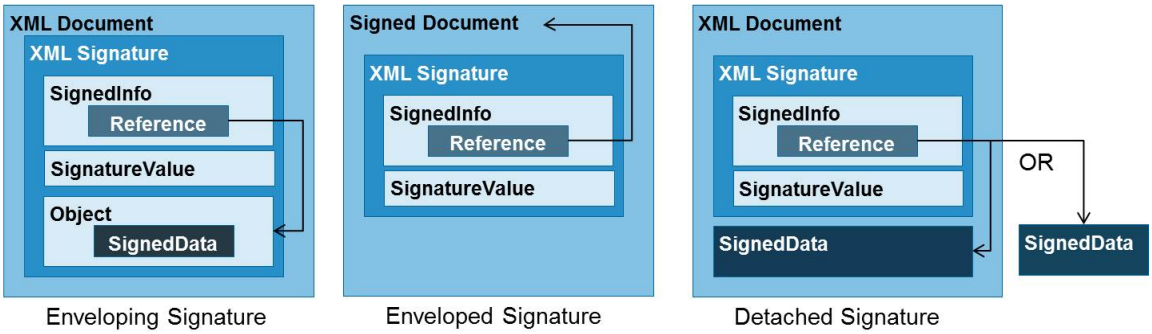


Figure 1 : XML Signature Structure

As depicted in Figure 1 , XMLDSig specifies three standard methods to associate a signature with the signed content:

- Enveloping Signature: An enveloping signature is the signature applied over the content found within an **Object** element of the signature itself. The **Signature** element is a parent to the signed element. The object or its content is identified through a **Reference** element by way of a Uniform Resource Identifier (URI) fragment identifier or transform.

- Enveloped Signature: An enveloped signature is the signature applied over the XML content that contains the signature as an element. The **Signature** element is the child to the signed content, which is identified by way of a *URI* with the use of enveloped transform. The **Signature** element is excluded from the calculation of the signature value.

- Detached Signature: A detached signature is the signature applied over the content external to the **Signature** element, and it can be identified by way of a URI or a transform. The signed XML resource may point to content contained within the same document, to content contained within an external resource, or to an external resource (signature applied across entire resource). The **Signature** element is neither a parent (enveloping signature) nor child (enveloped signature) to the signed content. The signature and the signed content may be sibling elements or be separate data objects. A detached signature is useful when you can't modify the source data to be signed.

XMLDSig uses an indirect signing mechanism that encrypts the hash of the reference data with transforms. Table 5 shows the algorithms specified by XMLDSig and how they are related with the XML signature syntax.

Table 5 - XML Signature Algorithms

Type	Algorithms Defined	Elements Defined
Digest	SHA 1, SHA 256, SHA 224, SHA 384, SHA 512	**ds:SignedInfo/Reference/ DigestMethod**
Encoding	base64	**ds:SignedInfo/Reference/DigestValue**
MAC	HMAC - SHA 1/224/256/384/512	**ds:SignedInfo/SignatureMethod**
Signature	RSA , ECDSA , DSA with SHA 1/224/256/384/512	**ds:SignedInfo/SignatureMethod**

Type	Algorithms Defined	Elements Defined
Canonicalization	Canonical XML 1.0/1.1 (with or omit comments), Exclusive XML Canonicalization 1.0 (with or omit comments)	**ds:SignedInfo/ CanonicalizationMethod**
Transform	Base64, Enveloped Signature, XPath , XPath Filter 2.0, XSLT	**ds:SignedInfo/Reference/Transform**, **ds:KeyInfo/RetrievalMethod/ Transform**

2.1.2 - Introduction to XML Encryption

The W3C Recommendation: *XML Encryption Syntax and Processing* (abbreviated in this document as XMLEnc) specifies XML encryption processing rules, syntax and algorithms. XMLEnc protects the privacy of the full XML document, an XML element, or the content of an XML element. The **EncryptedData** element is the core element that may be the root of a new XML document or a child element in an XML document. It contains the encrypted data, i.e. the **cipherData** element. As depicted in Figure 2 , the **cipherData** element may contain the encrypted cipher value, or provide a reference to an external location containing the encrypted cipher value via the **CipherReference** element.

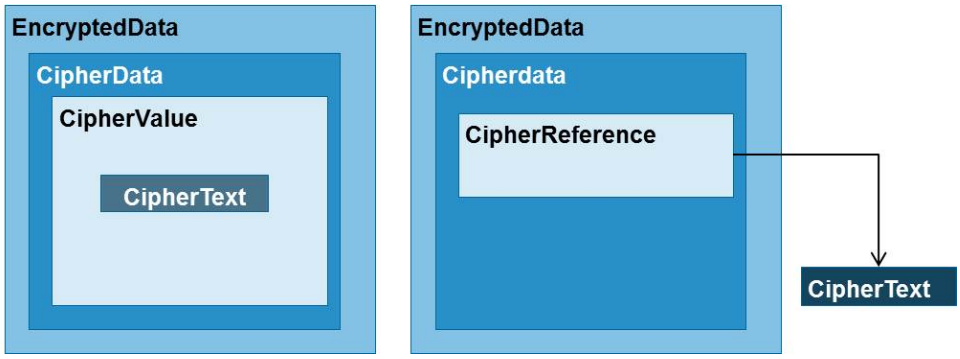


Figure 2 : XML Encryption Structure

EncryptionMethod is an optional element that describes the encryption algorithms applied to the cipher data. XMLEnc also defines the extensions to **ds:keyInfo** for keying materials needed to decipher the cypher data.

Table 6 shows the algorithms specified by XMLEnc and how they are related with the XML encryption syntax.

Table 6 - XML Encryption Algorithms

Type	Algorithms Defined	Elements Defined
Block Encryption	TRIPLEDES , AES128/192/256-CBC, AES128/192/256-GCM	**xenc:EncryptionMethod**
Key Derivation	ConcatKDF , PBKDF 2	**xenc11:DerivedKey/KeyDerivationMethod**
Key Transport	RSA - OAEP , RSA -v1.5	**ds:KeyInfo/xenc:EncryptedKey/xenc:EncryptionMethod**
Key Agreement	Elliptic Curve Diffie-Hellman, Diffie-Hellman Key Agreement (Legacy or explicit key derivation)	**ds:KeyInfo/xenc:AgreementMethod**
Symmetric Key Wrap	TRIPLEDES , AES -128/192/256	**ds:KeyInfo/xenc:EncryptedKey/xenc:EncryptionMethod**
Encoding	Base64	**xenc:CipherData/CipherValue**
Transforms	Base64, XPath , XPath Filter 2.0, XSLT	**xenc:CipherData/CipherReference/Transform, ds:KeyInfo/RetrievalMethod/Transforms**

2.2 - High Level Use Cases

XML was designed to transport and store data. XML signature and XML encryption are used to secure the full or the partial XML data. Different types of data impose different requirements on the security aspects and the security levels. For example, some data carries private information and confidentiality is required for the exchange of such data. Some data carries access control information so data integrity and authenticity are required. This guidance document focuses on data in transit through HTTP based web services and two special types of data are identified, security token and cryptographic binding information.

- Security token exchange has high level of security risks due to the characteristics of the security token data.
- Cryptographic binding information is constructed using multiple data sources and it is used to protect the data relationship.

In this section, three high level use cases for XML signature and XML encryption are described: security token exchange, cryptographic binding and information exchange. The first two use cases describe the use of two special types of data and information exchange is the use case

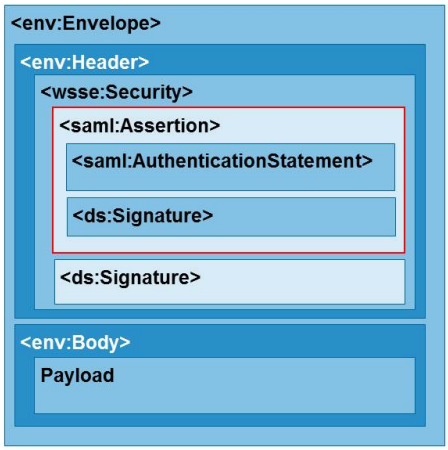
for XML data in general. The security requirements are also dependent on the specific application and the environment. Three different implementation scenarios in HTTP based web services are discussed later in Chapter 4 - Web Services Implementation Guidance .

The examples which appear in this section are only intended to illustrate one example scenario of the high level use case being specified. The examples are NOT intended to necessarily represent best practice for implementing the data flow of the particular application.

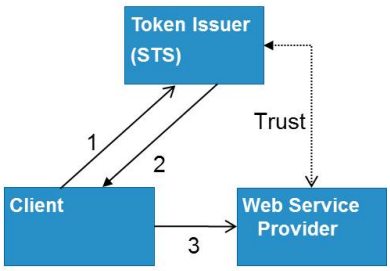
2.2.1 - Security Token Exchange

A security token represents a collection (one or more) of claims and it is used for access control. Security tokens may contain claims about a subject's authentication, a subject's authorization credentials, an authorization decision, or a combination of all. A signed security token is a security token that is asserted and cryptographically signed by a specific authority. A signed security token may be of binary type (e.g., X.509 certificate or a Kerberos ticket) or of XML type. A security token likely contains private data or secret and the token may pass through intermediaries in its transmission path therefore there is the need to protect the privacy of the token data. For an XML based security token, XMLDSig may be used to sign the token and XMLEnc may be used to encrypt the token.

Part (a) of Figure 3 shows an example security token. A signed Security Assertions Markup Language (SAML) token is embedded in the WS -Security header. The SAML Assertion contains the authentication statement and the signature.



(a) Example Security Token



b) Security Token Exchange

Figure 3 : Security Token Exchange

Part (b) of Figure 3 shows an example security token exchange scenario with the presence of token service. The token issuer, also called the Security Token Service (STS) , is trusted by the service provider and is capable of authenticating the clients. In order for the client to access the service, the client's request must contain a security token issued by the token issuer.

- 1. The client sends the token issuer a request for security token. The request is signed with an x.509 certificate that identifies the client and the token issuer can authenticate.

- 2. After the client is authenticated, the token issuer responds with a security token and its signature. The security token contains the authentication assertion and the signature of the assertion.

- 3. The client sends an invocation request to the server and embeds the security token in the request. When the service receives the request from the client, it verifies the security token first and then processes the message.

2.2.2 - Cryptographic Binding

Cryptographic Binding is a methodology for providing integrity and authenticity to data and the associated metadata. The metadata may be in any discrete format and it could be embedded or a separate file in a different repository. Multiple metadata files may be bound, e.g., discovery metadata, Information Assurance (IA) metadata, etc. Part (a) of Figure 4 shows that the binding information is created and signed using both the data asset and the metadata by cryptographic methods. Cryptographic binding process will not modify the data asset or metadata. When using XML encoding, XMLDSig may be used to sign the binding information. The XML signature may be formed over the concatenation of the data asset and the metadata, or the hashes of both. The XML signature and the binding info may be stored in a distinct file or be included in the metadata XML document. The validation of the signature will verify the combination of the data and the metadata.

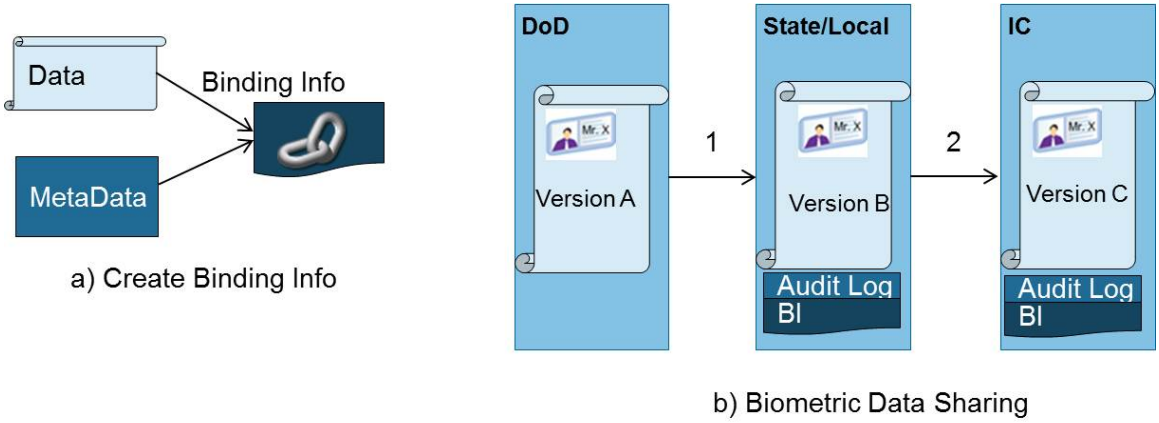


Figure 4 : Cryptographic Binding

Part (b) of Figure 4 shows an example cryptographic binding scenario for biometric data sharing. A biometric record is accessed and modified by different entities: DOD , State and IC . With each update, an audit log entry is created and the binding info (BI) data is computed over the data record and the auditing log entry.

- 1. The record of version A is sent to State and then updated to version B. The change history is logged in the audit log and the BI data for this log entry is created. The *BI* data contains the hashes of the recorded change and the record. The XML signature is formed over the *BI* data and included as a detached signature in the *BI* data.

- 2. The record of version B, the audit log and the BI data are sent to IC . To validate the received data, the XML signature is first verified to ensure authenticity and that the hash

values were not changed. Next the hash values of the received audit log and the record are compared against the hash values in the *BI* data. If the hashes match then the asset has integrity. Another update is performed by IC and a record of version C is created. The change history is logged in the audit log and the BI data for this log entry is added. More details of this binding scenario is found in [16].

When the authenticity and integrity of the auditing data is protected by cryptographic binding, it is possible to detect modifications, insertions, deletions, or unauthorized data sources and to facilitate assured synchronization of data and data collections.

2.2.3 - Information Exchange

For general information other than security token and data relationship, XMLDSig and XMLEnc can be used to secure the data when XML encoded information is exchanged among multiple entities and endpoints. Message level authenticity, integrity, confidentiality and fine-grained access are desired if a message must first pass through intermediaries before reaching its final destination, or the message may be stored in database, in cache or be saved as a file for later retrieval, or the environment is open (e.g., cloud).

Part (a) of Figure 5 shows that an XML document is secured by signing and encryption.

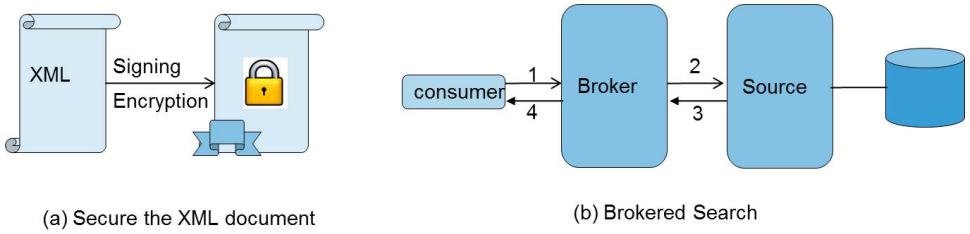


Figure 5 : Information Exchange

Part (b) of Figure 5 shows an example information exchange scenario for broked search. The broker, i.e. the brokered search service provider, distributes the search request to different sources, i.e. the search service providers. The broker acts as an intermediary and it is possible that the consumer and the source are authorized a higher level of access than the broker.

- 1. The consumer sends a search request to the broker. The search request may contain the security assertions on the consumer's identity and attributes.
- 2. The broker directs the search request to the source. The consumer's identity and attributes may be embedded in the request and passed to the source or there may be back channels for the source to obtain consumer information and the authorization decision.
- 3. The source sends the search results in XML format to the broker.
- 4. The broker processes the search results and sends the XML formatted results to the consumer.

To protect sensitive information in transit such that it is available only to those who are authorized to access it, the use of signature and encryption is required in order to maintain the

authenticity, confidentiality and integrity of the data as it is sent from the source to the destination, e.g., from the source to the consumer in the brokered search example.

2.3 - Data Standards

This section identifies and introduces the key specifications that are related with the use cases described in Section 2.2.

2.3.1 - SAML

Security Assertions Markup Language (SAML) [22] is an XML -based framework for communicating user authentication, authorization, and attribute information. It allows assertion entities (e.g., the identity providers) to make assertions to the other entities (e.g., the service providers), regarding the identity, attributes, and authorization of a subject (i.e. the relying entity) that is often a human user. SAML tokens are XML representations of claims and carry statements that are sets of claims made by the assertion party about the relying entity. SAML specifies the use of the XMLDSig to sign the SAML token and the use of XMLEnc to protect sensitive data in the SAML token.

2.3.2 - XACML

The OASIS Extensible Access Control Markup Language (XACML) [27] specifies schemas for authorization policies and for authorization decision requests and responses. The SAML profile of XACML describes how an instance of an XACML authorization statement is enclosed in a SAML assertion and the assertion is used as an XACML authorization token.

An important geo-specific extension to XACML is defined by Geospatial Extensible Access Control Markup Language (GeoXACML)[28] to specify access control policies to Geo-Spatial data.

2.3.3 - ATOM

Atom[13] is an XML -based document format that describes lists of related information known as “feeds” and it is used for information exchange. Various specifications leverage the Atom syndication format and guidelines are developed to ensure consistent usage across those specifications. For example, Atom Data Encoding Specification for Content Discovery and Retrieval (CDR) Result Sets [1]is developed to support the specific information requirements of the CDR Search Specifications.

2.3.4 - WS-Security

Web Services Security (WS-Security) [26] defines how to apply XMLDSig and XMLEnc to the body of a SOAP message. It is used for SOAP based information exchange and it also addresses how to bind various security tokens to ascertain the sender’s identify. Three token profiles are developed: X.509 Token profile, Kerberos Token profile and SAML token profile. The token profiles describe how to embed the security token in the WS -Security header and how to sign the security token.

2.3.5 - Cryptographic Binding Methods

Cryptographic binding techniques are used by various services and applications. For example, the cryptographic binding CONOPS [17] defines the binding and validation services and specifies the use of XMLDSig as one cryptographic binding method; IA Type 98 Data Best Practices[16] specifies the guidelines for creation and processing of XMLDSig based cryptographic binding information.

The IC Trusted Data Format (IC-TDF.XML) specification[5] defines detailed implementation guidance for using Extensible Markup Language (XML) to encode IC-TDF.XML data. A key concept in the IC-TDF.XML specification is the ability to cryptographically assure the relationship among portions of the document.

Chapter 3 - Guidance for Use of XML Signature and XML Encryption

3.1 - Known Vulnerabilities

Proper use of XMLDSig and XMLEnc requires a thorough understanding of the semantics and the processing rules. Various types of attacks and evasions against XMLDSig and XMLEnc have been identified. Attacks may be related with the syntax of XMLDSig and XMLEnc , and they fall into these categories:[31]

- Transform injection by XPath or XSLT through complex transform and custom code execution.

  As Table 5 and Table 6 show, transform algorithms are used by XMLDSig and XMLEnc for references and key retrieval methods. Complex XSLT transform or expensive XPath expression might lead to a denial of service (DOS) attack. User-defined extensions might execute a risky program in the operating system (OS) .

- Reference related attacks through unsafe remote content and element wrapping.

  As Table 5 and Table 6 show, transform algorithms are used by XMLDSig for signed info and by XMLEnc for cipher data. External references to the file system or other web sites can cause exceptions or cross site attacks. With element wrapping attack, modifications may be made to unprotected content or signed elements get moved to different places.

- C14N (Canonical XML) related attacks through excessive transform, entity expansion and hash collision.

  As Table 5 shows, C14N algorithms are used by XMLDSig for canonicalization. Expensive C14N transform might lead to a denial of service attack. When using C14N with comments, there are risks of hash collision if allowing comments in the signed info.

The attacks may also be related with XMLDSig and XMLEnc processing and the algorithms in general. Here is a list of possible attacks:[31] [29]

- replay attack using element substitution and wrapping;
- use of error messages to reveal the details of algorithm implementation;
- timing attack against the encrypted key;
- plain-text guessing;
- unsafe keys;
- attacks to algorithms with known security risks.

3.2 - General Guidance

To mitigate the security risks associated with XMLDSig and XMLEnc , recommendations are made in this section, based on the XMLDSig and XMLEnc specifications,[29] [30] the best

practice guide document from W3C [31] and publications from the National Institute of Standards Technology (NIST) .[15] [3]

3.2.1 - Consolidated Guidelines for General Use of XMLDSig and XMLEnc

In this section, high level recommendations are made for the general use of XMLDSig and XMLEnc . Under each high level recommendation, there are specific guidelines against different types of attacks. Some of the guidelines apply to both XMLDSig and XMLEnc because they apply to the common components and the common processes, e.g., use of transform, use of reference, and the key retrieval process. Some apply to XMLDSig only, e.g., the use of C14N canonicalization. Some guidelines apply to XMLEnc only, e.g., the use of encryption method.

3.2.1.1 - Establish Policies for Use of Transforms and References

Strict policies for the use of XSLT transform, XPath transform, C14N transform and reference methods can effectively prevent the XML document from those attacks that exploit complex, unsafe transform and references. The policy may limit the type of transforms, the number of transforms, the order of the transforms and the type of reference to be used. The policies are application specific and environmental specific. To implement the policies, coordination is required between the XML signature signer and the verifier for XMLDSig , or between the encrypting process and the decryption process for XMLEnc .

Table 7 lists the recommendations for use of transforms and references.

Table 7 - Best Practices For Use of Transforms and References

#	Type of Vulnerability	Recommendation	XMLDSig	XMLEnc
1.	DOS (arbitrary XSLT transforms)	MUST limit XSLT transform to trusted sources only; SHOULD NOT use or process XSLT transform for **ds:KeyInfo/ds:RetrievalMethod**; MAY disallow XSLT transforms; MAY define custom named transform to limit scope.	yes	yes
2.	DOS (user-defined XSLT transforms) Code Execution	MUST limit user-define extensions to trusted sources only; MAY disable user-defined extensions.	yes	yes

#	Type of Vulnerability	Recommendation	XMLDSig	XMLEnc
3.	DOS (Complex XPath transform)	MUST limit XPath transform to trusted sources only; SHOULD use XPath Filter 2.0; MAY accept limited set of XPath expressions, e.g., use ancestor, self axes only and don't compute string-value of elements; MAY disallow XPath transforms.	yes	yes
4.	DOS (wildcard XPath selection)	MUST limit streaming-based XPath selection to trusted sources only; SHOULD limit the use of wildcard XPath selection, e.g., avoid using the "descendant", "descendant-or-self", "following-sibling", and "following" axes when using streaming XPaths .	yes	yes
5.	DOS (**ds:RetrivalMethod**)	SHOULD limit the use of **ds:RetrievalMethod** with **ds:KeyInfo**, e.g., same-document URI reference only, and the transform allowed; MAY disallow using **ds:RetrievalMethod**.	yes	yes
6.	DOS (external reference)	SHOULD limit the size and timeout values for content retrieved over the network; SHOULD provide cached reference to the verified content; SHOULD limit external reference to trusted reference, e.g., use "cid:" URL in Web Service Security for accessing attachment; MAY constrain outbound network connectivity from XSLT processor; MAY disallow using external reference.	yes	yes
7.	DOS (excessive transforms)	SHOULD limit the number of transforms allowed in a transformation chain;	yes	yes
8.	DOS (C14N Entity expansion)	SHOULD NOT transmit unparsed external entity references in signed material; SHOULD expand all entity references before creating the clear text that is transmitted.	yes	no

#	Type of Vulnerability	Recommendation	XMLD Sig	XMLEnc
9.	plain-text guessing attacks (Combining signature and encryption)	SHOULD encrypt any digest or signature over that data when data is encrypted; MAY employ the “decrypt-except” signature transform.	yes	yes

3.2.1.2 - Use the Recommended Algorithms and the Recommended Configuration

There are known vulnerabilities with some cryptographic algorithms and the use of such algorithms should be avoided. For example, cipher-block chaining (CBC) block encryption algorithms should not be used due to possibly severe security risks. SHA -1 is discouraged due to concerns on long-term collision resistance. There are known attacks against the encrypted key when using PKCS #1.5 algorithm. Section 3.2.2 - Consolidated Guidelines for Cryptographic Algorithms discusses the security strength of the algorithms and recommends the algorithms to use and not to use. For the selected algorithm, proper configuration is required and the recommendations for algorithm configuration are shown in Table 8 .

Table 8 - Best Practices For Algorithm Configuration

#	Type of Vulnerability	Recommendation	XMLD Sig	XMLEnc
1.	Signature Evasion (Hash collision, forgery and key recovery)	MUST set RSA minimum key size 2048; MUST set ECDSA minimum f=256; MUST NOT use SHA 1 with ConcatKDF ; MUST NOT use SHA 1 with PBKDF ; SHOULD set the HMAC output length to one half the number of bits in the hash size; SHOULD use minimum SHA 256 with ConcatKDF ; SHOULD use minimum SHA 256 with PBKDF .	yes	yes

#	Type of Vulnerability	Recommendation	XMLD Sig	XMLE nc
2.	Attacks against the encrypted key	MUST NOT re-use IV(Initialization Value or Vector); SHOULD control the generation of IV and protect the IV till the end of its use as critical security parameter (CSP) being defined by FIPS 140-2;[2] SHOULD use random IV; SHOULD use the proper framework to construct IV with AES - GCM [14], i.e. use deterministic construction framework for IV length strictly less than 96bits; use either deterministic construction framework or RBG -based construction framework for IV length 96bits or greater.	no	yes

3.2.1.3 - Keep Key Safe

It is critical to keep key safe in order to secure the XML signature process and the encryption process.

- Trust MUST be established before other potentially risky operations.
- There are known attacks against encrypted key to break the security by exploiting the vulnerabilities in some encryption algorithms.
- To keep the key safe, it is important to use distinct keys for different purposes.

The recommended practices for the use of keys are listed in Table 9 .

Table 9 - Best Practices For Safe Key

#	Type of Vulnerability	Recommendation	XMLD Sig	XMLE nc
1.	DOS (Unsafe key) Signature evasion	MUST validate X.509 certificates, certificate chains and revocation status.	yes	yes
2.	Attacks against the encrypted key (PKCS #1.5)	SHOULD use RSA - OAEP ; MAY generate a random secret key every time when the decrypted data was not PKCS #1-conformant.	no	yes

#	Type of Vulnerability	Recommendation	XMLD Sig	XMLEnc
3.	Backwards Compatibility Attacks (Eavesdropped Legacy Algorithms)	SHOULD always use a different public key pair for data confidentiality and for data integrity functionality; SHOULD NOT use the same key material for different algorithms for symmetric keys, even if serving the same purpose; SHOULD use RSA - OAEP , AES - GCM ; MAY reject documents containing RSA - PKCS #1 v1.5 and AES - CBC ciphertexts without decryption.	no	yes
4.	Information Revealed	SHOULD use the symmetric key only to the data intended for all recipients when the key is shared amongst multiple recipients.	no	yes
5.	Signature Forgery	SHOULD use distinct keys when encrypting and signing; SHOULD use key derivation to produce different keys when using a single key.	yes	yes

3.2.1.4 - Follow Recommended Processing Order and Rules

Additional rules during XML signature processing and encryption processing are listed in Table 10 .

Table 10 - Best Practices For Processing Orders and Rules

#	Type of Vulnerability	Recommendation	XMLD Sig	XMLEnc
1.	DOS (Reference Validation)	SHOULD follow this order of operations to verify/validate the signature: 1. Fetch the verification key and establish trust in that key *2*. validate**ds:SignedInfo** with that key *3*. validate the references	yes	no
2.	Signature Evasion (Reference Element Wrapping)	MUST check both the name and position of an element as part of signature verification.	yes	no

#	Type of Vulnerability	Recommendation	XMLD Sig	XMLEnc
3.	Replay Attack	SHOULD sign all parts of the document SHOULD include user names, keys, timestamps, etc. into the signature; SHOULD use a nonce in combination with signing time; MUST include nonce and signing time into the signature; SHOULD detect replay at the security processing layer and SHOULD NOT rely on application logic since applications may change.	yes	no
4.	Unsafe Content (executable code, viruses, etc.).	MUST ensure that arbitrary content can be safely processed by receiving applications; SHOULD inspect the decrypted contents; MAY limit access to decrypted contents.	no	yes
5.	Plain-text Guessing	SHOULD encrypt any digest or signature over that data when data is encrypted.	yes	yes
6.	Information Revealed	SHOULD NOT provide detailed error responses related to security algorithm processing; SHOULD limit error messages to a generic error message. SHOULD not reveal any information in parameters or algorithm identifiers (e.g., information in a URI) that weakens the encryption, e.g., use generic terms in parameter names or identifier names to avoid inappropriate disclosure of system and application information.	yes	yes

3.2.1.5 - Limit Processing and Network Resource

The XML signature validation and decryption process may limit the total amount of processing and networking resources that a request can consume. This is one way to mitigate DOS attack such that a malicious message would not bring down an entire set of web applications and services. The recommended practices are listed in Table 11 .

Table 11 - Best Practices For Resource Restriction

#	Type of Vulnerability	Recommendation	XMLD Sig	XMLEnc
1.	DOS (external reference)	MAY constrain outbound network connectivity from XSLT processor	yes	yes

#	Type of Vulnerability	Recommendation	XMLDSig	XMLEnc
7.	DOS (excessive transforms)	SHOULD limit the total amount of processing and networking resources a process can consume.	yes	yes
2.	Timing Attacks	SHOULD ensure that distinct errors detected during security algorithm processing do not consume systematically different amounts of processing time from each other; SHOULD treat as suspect inputs when a large number of security algorithm processing errors are detected within a short period of time, especially in messages from the same origin.	no	yes

3.2.2 - Consolidated Guidelines for Cryptographic Algorithms

Recommendations on the use of algorithms are based on the requirements specified by XMLEnc and XMLDSig specifications and the algorithm strength reported by NIST publications.

The recommended algorithms all provide sufficient level of security. According to NIST-SP800-57[15], the strength of the cryptographic algorithm is measured by “bits of security”, which is a number associated with the amount of work (that is, the number of operations) that is required to break a cryptographic algorithm or system. Security life of the data is defined as the time period during which the security of the data needs to be protected (e.g., its confidentiality, integrity or availability). Security-strength time frames specify during which the security strength is either acceptable, disallowed, deprecated or OK for legacy use for applying and processing. Appendix E - NIST Algorithm analysis lists the algorithm analysis results from NIST-SP800-57.

In order to maximize interoperability, the recommended algorithms may not provide the highest level of security. XMLDSig and XMLEnc specify three different levels of implementation support: “Required”, “Optional”, “Recommended”. The support for all of the recommended algorithms is at the level of “required” for better interoperability.

Table 12 lists the algorithms recommended for XML signature. The value of bits of security is used to describe the level of strength. When bits of security are not applicable for one type of algorithm, e.g., the canonicalization algorithms, known vulnerabilities to security attacks are used to describe the security strength. The value of security strength time frame is also used to decribe the level of strength. If the use of algorithm is disallowed beyond a security time frame of 2031, its security time frame value is included in the table. Otherwise, the use of algorithm is acceptable beyond 2031 by default.

Table 12 - Recommended Algorithms for XML Signature

Type	Algorithm (URI)	Level of Security
Digest	SHA 256 *http://www.w3.org/2001/04/xmlenc#sha256*	128

Type	Algorithm (URI)	Level of Security
MAC	HMAC - SHA 256 *http://www.w3.org/2001/04/xmldsig-more#hmac-sha256*	256
Signature [a]	RSAwithSHA256 for better interoperability and fast verification [b] *http://www.w3.org/2001/04/xmldsig-more#rsa-sha256* Or ECDSAwithSHA256 for small size keys and small size signatures *http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256*	128
Canonicalization	Exclusive Canonical XML 1.0 (omit comments) *http://www.w3.org/2001/10/xml-exc-c14n#*	Possible DOS attack
Encoding	base64 *http://www.w3.org/2000/09/xmldsig#base64*	None[c]

[a]Note that the MAC algorithms and the digital signature algorithms are both used to generate the signature but the MAC algorithms use a shared secret key while the digital signature algorithms are public key based.

[b]Both RSA based and ECDSA based signature algorithms provide sufficient level of the security. RSA algorithms are well established and widely supported. It has better verification performance than ECDSA (http://nicj.net/files/performance_comparison_of_elliptic_curve_and_rsa_digital_signatures.pdf). ECDSA may be preferred if smaller size keys and signatures are desired. Both ECDSA signatures and public keys are much smaller than RSA signatures and public keys of similar security levels. The application should choose the algorithm based on its system requirements. Data standards specific recommendations on signature algorithms are provided in Section 3.3 - Data Standards Guidance.

[c]The base64 algorithm encodes binary data to printable text. It may be unreadable by the naked eye however it can be decoded with minimum effort.

Table 13 lists the algorithms recommended for XML encryption.

Table 13 - Recommended Algorithms for XML Encryption

Type	Algorithm (URI)	Level of Security
Block Encryption	AES -128- GCM *http://www.w3.org/2009/xmlenc11#aes128-gcm*	128
Key Derivation	ConcatKDF using SHA 256 *http://www.w3.org/2009/xmlenc11#ConcatKDF*	256 with use of SHA 256

Type	Algorithm (URI)	Level of Security
Key Transport	RSA - OAEP (including MGF1 with SHA 1) *http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p* minimum key size=2048	112 with k=2048
Key Agreement	Elliptic Curve Diffie-Hellman (Ephemeral-Static mode) *http://www.w3.org/2009/xmlenc11#ECDH-ES*	128[a]
Symmetric Key Wrap	AES -128 Key Wrap *http://www.w3.org/2001/04/xmlenc#kw-aes128*	128
Message Digest	SHA 256 *http://www.w3.org/2001/04/xmlenc#sha256*	128
Encoding	base64 *http://www.w3.org/2000/09/xmldsig#base64*	None[b]

[a]The value for bits of security is not reported for the algorithm and it is derived based on comparable performance with ECDSAwithSHA256 and the use of ConcatKDF .[3]

[b]The base64 algorithm encodes binary data to printable text. It may be unreadable by the naked eye however it can be decoded with minimum effort.

There are algorithms with known vulnerabilities and those algorithms MUST NOT be used:

Table 14 - Algorithms NOT to Use

Type	Algorithm (URI)	Level of Security
Digest	SHA 1 *http://www.w3.org/2000/09/xmldsig#sha1*	80 Deprecated by 2013; Disallowed to apply and legacy use only beyond 2014
Signature	DSAwithSHA1 *http://www.w3.org/2000/09/xmldsig#dsa-sha1*	80 Deprecated by 2013; Disallowed to apply and legacy use only beyond 2014
Signature	RSAwithSHA1 *http://www.w3.org/2000/09/xmldsig#rsa-sha1*	80 Deprecated by 2013; Disallowed to apply and legacy use only beyond 2014

Type	Algorithm (URI)	Level of Security
Signature	ECDSAwithSHA1 *http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha1*	80 Deprecated by 2013; Disallowed to apply and legacy use only beyond 2014
Key Transport	RSA -v1.5 *http://www.w3.org/2001/04/xmlenc#rsa-1_5*	PKCS #1.5 attacks

A complete list of recommendations on all algorithms is provided in Appendix C - Complete Guide for All XMLDSig and XMLEnc Defined Algorithms .

3.3 - Data Standards Guidance

This section specifies the best practices for the specifications identified in Section 2.3 - Data Standards . The recommendations are based on the specifications and other findings from the industry.

3.3.1 - SAML

SAML 2.0[22]specifies the use of XML digital signatures and encryption in the SAML request and response message. Multiple SAML protocol bindings for the use of SAML messages are defined[SAML 2.0 binding]. The protocol binding maps SAML request-response message exchanges onto standard messaging or communication protocols, e.g., SOAP , HTTP and URI . The security requirements for SAML message authentication, integrity and confidentiality may vary with the protocol binding being used:

- The SAML messages may pass through a SOAP intermediary with reverse SOAP (PAOS) binding.
- The SAML message may pass through the user agent intermediary, e.g., a browser, with HTTP redirect binding, HTTP POST binding, HTTP Artifact binding.

SAML specific XML signature and encryption processing rules are required in addition to the general guidelines in Section 3.2 - General Guidance . Firstly, the use of XMLDSig and XMLEnc is specified in the context of the specific protocol exchange and the deployment environment.

- The SAML request and response MUST be bi-directionally authenticated when a relying party (e.g., a human user or a web application) requests an assertion from an asserting party (e.g., STS). Transport Layer Security (SSL 3.0 or TLS 1.0) using server and client authentication or authentication via digital signatures SHOULD be used.
- XMLDSig and XMLEnc SHOULD be used at the SOAP message layer with PAOS binding for end to end security when the HTTP requester in PAOS binding (e.g., web application) acts a SOAP intermediary.

- When a response message containing an assertion is delivered to a relying party via a user's web browser (e.g., using the HTTP POST binding), the response message MUST be digitally signed using XML Signature to ensure message integrity.

For the use of XMLDsig, SAML2.0 specifies the optional **ds:Signature** elements within SAML assertions, requests, and responses. XML Signature Profile is defined and it details the constraints on the use of XMLDSig . [1]

- SAML assertions and protocols MUST use enveloped signatures when signing assertions and protocol messages.

- The XML Signature in SAML message SHOULD NOT contain transforms other than the enveloped signature transform. Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid.

- SAML assertions and protocol messages MUST supply a value for the **ID** attribute on the root element of the assertion or protocol message being signed. The signatures MUST contain a single **ds:Reference** containing a same-document reference to the **ID** attribute value of the root element of the assertion or protocol message being signed.

SAML assertions will likely contain private data and the assertions may be passed around among different parties and may be stored in a cache or in a database. Confidentiality may also be required to protect the holder-of-key **ds:SubjectConfirmation** secret. The SAML schema is defined to be compatible with the inclusion of the encrypted data. The following recommendations on the use of XMLEnc are made:

- The **ds:BaseID**, **ds:NameID**, **ds:Attribute** elements and **ds:SubjectConfirmation** secret MAY be encrypted by use of XMLEnc .

- Additional guidelines to key and data referencing are provided in SAML specification to facilitate interoperability.

Recent attacks on SAML [2] found vulnerabilities with various public SAML libraries. Application developers SHOULD use the latest version of the SAML libraries. For example, sophisticated element wrap attack affected OpenAM and OpenSAML (used by CAS). Security fixes were provided to OpenSAML in Java, V2.5.0 above and OpenAM 9.5.4 above.

3.3.2 - XACML

XACML specification [27] defines the XACML digital signature profile. It RECOMMENDS the use of XACML schema instances in SAML Assertions, Requests, and Responses, which may then be digitally signed as specified in the SAML specification. Therefore, recommendations on use of XMLDsig and XMLEnc in SAML SHOULD be applied to XACML .

[1] Some guidelines in the SAML specification are removed due to conflict with the latest findings in cryptographic, e.g., recommendation on the use of RSA - SHA 1.

[2] See USENIX Security 2012 paper from J. Somorovsky etc "On Breaking SAML: Be Whoever You Want to Be" available online at https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final91.pdf

3.3.3 - ATOM

ATOM specification[13] specifies the use of XML digital signatures and encryption in ATOM. These guidelines are in addition to the general guidelines in Section 3.2 - General Guidance and they provide ATOM specific XML signature and encryption processing rules.

On the use of XML signature, ATOM specifies the contents to sign and not to sign. Knowing what is signed protects the ATOM document from those attacks that exploit the schema and alter the contents, e.g., element wrapping attack.

- The root of an ATOM document (i.e., **atom:feed** in an Atom Feed Document, **atom:entry** in an Atom Entry Document) or any **atom:entry** element MAY be signed with an enveloped signature.
- Other elements in an ATOM document MUST NOT be signed unless their definitions explicitly specify such a capability.
- A source element SHOULD be added before signing if an entry does not contain its own **atom:source** element.
- Atom documents SHOULD NOT use MACs for signatures due to security issues with potential keying material handling issues.

The following recommendation on the use of XML encryption is made: [3]

- The root of an Atom document (i.e., **atom:feed** in an Atom Feed document, atom:entry in an Atom Entry document) MAY be encrypted

Guidelines to both signing and encrypting include:

- When an Atom document is to be both signed and encrypted, the document SHOULD first be signed and then encrypted.
- If MACs are used for authentication, the order MUST be that the document is signed and then encrypted.

3.3.4 - WS-Security

WS -Security [26] specifies the use of XML digital signatures and XML encryption in SOAP . Multiple token profiles are defined, i.e. username token, SAML token, X.509 token and kerberos token. WS -Security also defines the global id **wsu:Id** and a security timestamp **wsu:timestamp** for security purpose. In addition to the general guidelines in Section 3.2 - General Guidance , there are WS -Security specific XML signature and encryption processing rules, which are related with the processing of security token, timestamp, id reference and SOAP . Firstly, the contents to sign are specified:

- IDs and timestamp elements SHOULD be signed and the message recipients SHOULD discard (ignore) any message whose security semantics have passed their expiration.

[3]Some guidelines in the ATOM specification are removed due to conflict with the latest findings in cryptographic, e.g., recommendations on the use of AES -128 CBC .

- Messages SHOULD include digitally signed elements to allow message recipients to detect replays of the message when the messages are exchanged via an open network. Such elements can be part of the message or of the headers defined from other SOAP extensions. Four typical approaches are: Timestamp, Sequence Number, Expirations and Message Correlation.

- All relevant and immutable message content SHOULD be signed by the message producer. Receivers SHOULD only consider those portions of the document that are covered by the producer's signature as being subject to the security tokens in the message.

- Applications SHOULD sign the entire SOAP body.

- Security tokens appearing in **wsse:Security** header elements SHOULD be signed by their issuing authority so that message receivers can have confidence that the security tokens have not been forged or altered since their issuance.

- A message producer SHOULD sign any **wsse:SecurityToken** elements that it is confirming and that are not signed by their issuing authority

- The public key provided in the request SHOULD be included under the signature of the request.

In addition, there are recommendations on the use of transform, reference and security timestamp:

- An ID reference SHOULD be used instead of a more general transformation, especially XPath .

- References to elements with location-independent semantics SHOULD be used, for example, references using XPath transforms with Absolute Path expressions with checks performed by the receiver that the URI and Absolute Path XPath expression evaluate to the digested nodeset.

- Signed timestamps MAY be used to keep track of messages (possibly by caching the most recent timestamp from a specific service) and detect replays of previous messages. Timestamps SHOULD be cached for a given period of time, as a guideline, a value of five minutes can be used as a minimum to detect replays. Timestamps older than that given period of time SHOULD be rejected.

WS -Security leverages XMLEnc and allows encryption of any combination of body blocks, header blocks, and any of these sub-structures. Recommendations on the contents to encrypt are:

- The SOAP header blocks SHOULD be encrypted to preserve confidentiality.

- The value of **wsse:Password** in username token profile SHOULD be protected by either transport layer security or use of XMLEnc . The SOAP header blocks SHOULD be encrypted to preserve confidentiality.

- If the associated originating signature is received in encrypted form, then the corresponding **wsse11:SignatureConfirmation** element SHOULD be encrypted.

3.3.5 - Cryptographic Binding of Information Assurance Type 98 Data

The use of Cryptographic Binding (CB) is specified in the best practice guide to the ITL Information Assurance (IA) logical record (Type 98) data. [16] The processes to create and verify the binding data are defined. When using XML encoding, the Binding Information (BI) consists of XML Digital Signature with a manifest and signature block. A hash of the audit log is stored in the XML digital signature manifest. The manifest is used to allow for granularity in validation and verification. Recommendations are made on the signature block and the use of manifest:

- The signature block SHOULD include a reference with a hash to the binding attributes, image designation character, IA data creation date, and the manifest inside a **ds:SignedInfo** element.
- The manifest MAY use references not explicitly pertain to the Type 98 record (i.e. references to the other logical records) to allow “subset validation,” assuring that subsets of logical records ITL file have not been altered and that the hashes of the referenced data objects (e.g., logical records) have not been altered.

3.3.6 - Trusted Data Format (IC-TDF.XML)

The IC-TDF.XML specification[5] defines the optional **Binding** element on each **Assertion** and **HandlingAssertion** to assure the relationship among the assertion and the payload. The **Binding** element includes information about the algorithm and normalization method used to calculate **SignatureValue**. The signature is generated over the concatenated payload and assertion after normalization. IC-TDF.XML specific rules are applied to normalization and signing, e.g., normalization is applied only to the contents in **Statements**, **StatementMetadata** and **Payload**. The URI of the external reference, not the referenced object is normalized and signed.

IC-TDF.XML specification defines the **EncryptionInformation** element for the encryption of payloads, assertions, and keys. **EncryptionInformation** contains **KeyAccess** and **EncryptionMethod** with information necessary for decryption or key retrieval. Onion or layered encryption is possible with the use of **sequenceNum** attribute such that there will be multiple **EncryptionInformation** elements within one **EncryptionInformation** group. **EncryptionMethod** allows key size, algorithm, and Optimal Asymmetric Encryption Padding Scheme (OAEP) information.

Application developers may extend IC-TDF.XML to use XMLDSig and XMLEnc . [4] An enveloped XML signature may be used to provide the similar binding information for each **Assertion** or **HandlingAssertion**. The signature should be computed over the binding contents that IC-TDF.XML specifies. Rules and custom transforms may be defined to support IC-TDF.XML specific requirement as mentioned above. For encryption information, the encryption key size, algorithm and use of OAEP may be defined to meet IC-TDF.XML requirement.

[4]Please check the latest IC-TDF.XML specification on the support for XMLDSig and XMLEnc.

Chapter 4 - Web Services Implementation Guidance

This section provides guidance for the implementers to use XMLDSig and XMLEnc for XML message exchange in HTTP -based web services to meet the security requirement for the use cases as described in Section 2.2 - High Level Use Cases . HTTP -based web services may be implemented using various technologies and approaches, e.g., use of SOAP or REST as the web service interface, use of Transport Layer Security (SSL / TLS) as the additional layer of security underneath HTTP . The security requirements for web services are application specific and they are dependent on the deployment environment. Essentially the security requirement can be point-to-point or end-to-end:

- point-to-point security: secure message exchange between two endpoints with direct connection,
- end-to-end security: secure message exchange between two endpoints with intermediaries.

The guidelines in this section differentiate point-to-point security and end-to-end security for use of XMLDSig and XMLEnc with the implementation of SOAP web services, REST web services and the underlying SSL / TLS transport channel.

4.1 - SOAP Web services

SOAP is built on top of XML and it has many well-defined security standards such as:

- WS -Security for message authentication, integrity and confidentiality,
- WS -Trust for Security Token Service (STS) ,
- SOAP bindings of XML based security token exchange standards (SAML and XACML).

Recommendations on SOAP security for various use cases are found in Web Service Security High level Guide.[6]

Table 15 summarizes the recommendations for SOAP web service implementations.

Table 15 - SOAP Web Services

Use cases	Security Requirements	Recommendations
Security Token Exchange With token service	point-to-point	SAML 2.0 and WS -Trust SHOULD be used. Transport Layer Security (SSL / TLS) MUST be used. XMLDSig and XMLEnc May be used.
	end-to-end	SAML 2.0 and WS -Trust SHOULD be used. XMLDSig and XMLEnc SHOULD be used.

Use cases	Security Requirements	Recommendations
Security Token Exchange Without token service	point-to-point	WS -Security SAML Token Profile SHOULD be used. Transport Layer Security (SSL / TLS) or XMLDSig / XMLEnc SHOULD be used.
	end-to-end	WS -Security SAML Token Profile SHOULD be used. XMLDSig and XMLEnc SHOULD be used.
Information Exchange	point-to-point	Transport Layer Security (SSL / TLS) or WS -Security with XMLDSig / XMLEnc SHOULD be used.
	end-to-end	WS -Security with XMLDSig SHOULD be used for data integrity and authentication. WS -Security with XMLEnc SHOULD be used for confidentiality.
cryptographic binding	point-to-point	Transport Layer Security (SSL / TLS) SHOULD be used. BI data MAY be signed using XMLDSig if the binding information is sent in a SOAP message.
	end-to-end	BI data SHOULD be signed using XMLDSig if the binding information is sent in a SOAP message.

Following the recommendations above, the general guidelines and WS -Security, SAML specific guidelines apply to the use of XMLDSig and XMLEnc with SOAP web services. Additional standards specific guidelines should be applied if the application implements those standards.

4.2 - REST Web Services

REST web services mainly rely upon transport-layer security (SSL / TLS). Messages with REST web services use various different formats such as XML , JSON or plain text, etc. Security standards are emerging for REST web services:

- JSON Web Signature (JWS) for JSON based authentication and integrity
- JSON Web Encryption (JWE) for JSON based encryption
- JSON Web Token (JWT) as the security token
- OAuth for authorization
- OpenID for distributed authentication
- OpenIDConnect for both authentication and authorization

Recommendations on REST security for various use cases are found in Web Service Security High level Guide,[6] REST implementation guideline from the National Security Agency (NSA) on REST [18] and REST white paper from NSA .[19]

Table 16 summarizes the recommendations for RESTful web service implementations.

Table 16 - REST Web Services

Use cases	Security Requirements	Recommendations
Security Token Exchange With token service	point-to-point	SAML 2.0 and SAML 2.0 Web Browser SSO profile SHOULD be used. Transport Layer Security (SSL / TLS) MUST be used. XMLDSig / XMLEnc MAY be used.
	end-to-end	SAML 2.0 and SAML 2.0 Web Browser SSO profile SHOULD be used. XMLDSig and XMLEnc SHOULD be used.
Security Token Exchange Without token service	point-point	Mutually-authenticated Transport Layer Security (SSL / TLS) SHOULD be used.
	end-end	The REST Service Encoding Specification for End-to-End Identity Propagation (RR-ID [21]) SHOULD be used.
Information Exchange	point-to-point	Transport Layer Security (SSL / TLS) SHOULD be used. XMLDSig / XMLEnc MAY be used for XML based REST implementations.
	end-to-end	It is recommended to use the Atom Syndication Format to encapsulate the messages. Or the application may define a custom XML Schema which includes data type definitions from XMLDSig and XMLEnc . XMLDSig / XMLEnc SHOULD be used for XML based REST implementations.
cryptographic binding	point-to-point	Transport Layer Security (SSL / TLS) SHOULD be used. BI data MAY be signed using XMLDSig if the binding information is sent in XML .
	end-to-end	BI data SHOULD be signed using XMLDSig if the binding information is sent in XML . The similar recommendations for the use case of information exchange can be applied if the BI data is sent in an XML message.

Following the recommendations above, the general guidelines and ATOM, SAML specific guidelines should apply to the use of XMLDSig and XMLEnc with REST web services. Additional standard specific guidelines should be applied if the application implements those standards.

Note that JSON Web Signature (JWS) and JSON Web Encryption (JWE) may be used with REST implementations with JSON data structures. Some of the general guidelines in this guidance document can be applied to JWS and JWE , i.e., the recommended cryptographic algorithms to use and not to use.

4.3 - Use of Transport Layer Security (SSL/TLS)

Transport level security is underlying SOAP web service and REST web service and it is based on Secure Sockets Layer (SSL) or Transport Layer Security (TLS) that runs beneath HTTP . SSL / TLS provides security features including authentication, integrity and confidentiality for HTTP connections. Recommendations on SSL / TLS security for various use cases are found in the Web Service Security High level Guide.[6]

In general, use of SSL / TLS is recommended for point to point security, e.g., when there is a requirement to encrypt partial or all of the data between two points, with no intermediaries. Use of SSL / TLS is not sufficient for end to end security, e.g., when there is a requirement to encrypt some or all of the data between two points, with intermediaries. To meet end-to-end security requirement, the XML message SHOULD be signed for authentication/integrity and encrypted for confidentiality, regardless of whether it is transmitted by SSL / TLS . In this case, the general guidelines on use of XMLDSig and XMLEnc should be applied. When an application implements additional standards, the guidelines from those additional standards SHOULD be implemented.

Use of XMLDSig / XMLEnc with the underlying Transport Layer Security and the upper layer web service interfaces (SOAP or REST) must be considered together. The guidelines for use of XMLDSig and XMLEnc with Transport Layer Security are integrated in Table 15 and Table 16 for SOAP and RESTful web services, respectively.

Appendix A Feature Summary

The following table summarizes major features by version for WSS-SIGENC and all dependent specs. The “Required date” is the date when systems should support a feature based on the specified driver. For those changes driven by the IC Markings System Register and Manual, the date is often one year after the date of publication. Executive Orders, ISOO notices, ICD s and other policy documents have a variety of effective dates.

This document provides guidance on using W3C XML Signature and XML Encryption for XML message in transit though HTTP -based web service (e.g., SOAP and REST). It focuses on the vulnerabilities of XML signature and XML encryption and how to minimize the risks.

Table 17 - Feature Summary Legend

Key	Description
F	Full (able to comply and verified by spec to some degree)
P	Partial (Able to comply but not verifiable)
N	Non-compliance (Can’t comply)
N/A	Not Applicable. Feature is no longer required.
Cell Colors represent the same information as the Key value	

A.1. WSS-SIGENC Feature Comparison

Table 18 - WSS-SIGENC Feature comparison

WSS-SIGENC Feature Comparison		
Required date	**Feature**	**V1**
	Standards Specific guidelines for: SAML , XACML , WS -Security, Atom and cryptographic binding	F
	High level recommendations for the general use of XMLDSig and XMLEnc	F
	Implementation Specific guidelines for: SOAP , REST and the underlying Transport Layer Security (SSL / TLS)	F

Referenced work includes Open standards organizations (W3C , OASIS , NIST) and related DOD / IC specifications (Web Services Security Working Group (WSSWG) , NSA)

Appendix B Change History

The following table summarizes the version identifier history for this guidance document.

Table 19 - DES Version Identifier History

Version	Date	Purpose
1	14 March 2014	Initial Release

Appendix C Complete Guide for All XMLDSig and XMLEnc Defined Algorithms

Table 20 lists the complete set of algorithms defined by XMLDSig with the recommendations on the use, and the level of security for each algorithm. The algorithm name and URI are listed as well as the level of implementation support specified by XMLDSig .

Table 20 - Algorithms used by XML signature

Type	Algorithm (URI)	Implementation	Recommendation on Applying	Level of Security
Digest	SHA 1 http://www.w3.org/2000/09/xmldsig#sha1	Required only for backwards-compatibility reasons.	MUST NOT	80 Deprecated by 2013; Disallowed to apply and legacy use only beyond 2014
	SHA 256 http://www.w3.org/2001/04/xmlenc#sha256	Required	SHOULD	128
	SHA 224 http://www.w3.org/2001/04/xmldsig-more#sha224	Optional	MAY	112 Disallowed to apply and legacy use only beyond 2031
	SHA 384 http://www.w3.org/2001/04/xmldsig-more#sha384	Optional	MAY	192
	SHA 512 http://www.w3.org/2001/04/xmlenc#sha512	Optional	MAY	256
Encoding	based64 http://www.w3.org/2000/09/xmldsig#base64	Required	MUST	None

Type	Algorithm (URI)	Implementation	Recommendation on Applying	Level of Security
MAC	HMAC - SHA 1 http://www.w3.org/2000/09/xmldsig#hmac-sha1	Required	MUST NOT	128
	HMAC - SHA 256 http://www.w3.org/2001/04/xmldsig-more#hmac-sha256	Required	SHOULD	256
	HMAC - SHA 384 http://www.w3.org/2001/04/xmldsig-more#hmac-sha384	Recommended	RECOMMENDED	256+
	HMAC - SHA 512 http://www.w3.org/2001/04/xmldsig-more#hmac-sha512	Recommended	RECOMMENDED	256+
	HMAC - SHA 224 http://www.w3.org/2001/04/xmldsig-more#hmac-sha224	Optional	MAY	192
Signature	RSAwithSHA256 http://www.w3.org/2001/04/xmldsig-more#rsa-sha256	Required	SHOULD for fast verification performance and better interoperability	128
	ECDSAwithSHA256 http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256	Required	SHOULD for small size keys and small size signatures	128
	DSAwithSHA1 (signature verification) http://www.w3.org/2000/09/xmldsig#dsa-sha1	Required only for signature verification	MUST NOT	80 Deprecated by 2013; Disallowed to apply and legacy use only beyond 2014

Type	Algorithm (URI)	Implementation	Recommendation on Applying	Level of Security
	RSAwithSHA1 http://www.w3.org/2000/09/xmldsig#rsa-sha1	Recommended only for signature verification	MUST NOT	80 Deprecated by 2013; Disallowed to apply and legacy use only beyond 2014
	RSAwithSHA224 http://www.w3.org/2001/04/xmldsig-more#rsa-sha224	Optional	MAY	112 Disallowed to apply and legacy use only beyond 2031
	RSAwithSHA384 http://www.w3.org/2001/04/xmldsig-more#rsa-sha384	Optional	MAY	192
	RSAwithSHA512 http://www.w3.org/2001/04/xmldsig-more#rsa-sha512	Optional	MAY	256
	ECDSAwithSHA1 http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha1	Optional	MUST NOT	80 Deprecated by 2013; Disallowed to apply and legacy use only beyond 2014
	ECDSAwithSHA224 http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha224	Optional	MAY	112 Disallowed to apply and legacy use only beyond 2031
	ECDSAwithSHA384 http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha384	Optional	MAY	192
	ECDSAwithSHA512 http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512	Optional	MAY	256

Type	Algorithm (URI)	Implementation	Recommendation on Applying	Level of Security
	DSAwithSHA1 (signature generation) http://www.w3.org/2000/09/xmldsig#dsa-sha1	Optional for signature generation	MUST NOT	80 Deprecated by 2013; Disallowed to apply and legacy use only beyond 2014
	DSAwithSHA256 http://www.w3.org/2009/xmldsig11#dsa-sha256	Optional	MAY	128
Canonica-lization	Canonical XML 1.0 (omit comments) http://www.w3.org/TR/2001/REC-xml-c14n-20010315	Required	MAY	Possible DOS attack
	Canonical XML 1.1 (omit comments) http://www.w3.org/2006/12/xml-c14n11	Required	MAY	Possible DOS attack
	Exclusive XML Canonicalization 1.0 (omit comments) http://www.w3.org/2001/10/xml-exc-c14n#	Required	SHOULD	Possible DOS attack
	Canonical XML 1.0 (with comments) http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments	Recommended	MAY	Possible DOS attack
	Canonical XML 1.1 (with comments) http://www.w3.org/2006/12/xml-c14n11#WithComments	Recommended	MAY	Possible DOS attack

Type	Algorithm (URI)	Implementation	Recommendation on Applying	Level of Security
	Exclusive XML Canonicalization 1.0 (with comments) http://www.w3.org/2001/10/xml-exc-c14n#WithComments	Recommended	MAY	Possible DOS attack
Transform	base64 http://www.w3.org/2000/09/xmldsig#base64	Required	MAY	None
	Enveloped Signature http://www.w3.org/2000/09/xmldsig#enveloped-signature	Required	MUST for enveloped signature	None
	XPath http://www.w3.org/TR/1999/REC-xpath-19991116	Recommended	MAY	Possible DOS attack
	XPath Filter 2.0 http://www.w3.org/2002/06/xmldsig-filter2	Recommended	MAY	Possible DOS attack
	XSLT http://www.w3.org/TR/1999/REC-xslt-19991116	Optional	MAY	Possible DOS attack and Code Execution

Table 21 lists the complete set of algorithms used by XMLEnc with the recommendation on the use and the level of security for each algorithm. The algorithm name and URI are listed as well as the level of implementation support specified by XMLEnc

Table 21 - Algorithms used by XML Encryption

Type	Algorithm (URI)	Implementation	Recommendation on Applying	Level of Security
Block Encryption	TRIPLEDES http://www.w3.org/2001/04/xmlenc#tripledes-cbc	Required for backward compatibility	MAY use with additional layer of security such as SSL / TLS	112 Disallowed to apply and legacy use only beyond 2031 CBC plaintext Attack
	AES -128- CBC http://www.w3.org/2001/04/xmlenc#aes128-cbc	Required	MAY use with additional layer of security such as SSL / TLS	128 CBC plaintext Attack
	AES -256- CBC http://www.w3.org/2001/04/xmlenc#aes256-cbc	Required	MAY use with additional layer of security such as SSL / TLS	256 CBC plaintext Attack
	AES -128- GCM http://www.w3.org/2009/xmlenc11#aes128-gcm	Required	SHOULD	128
	AES -192- CBC http://www.w3.org/2001/04/xmlenc#aes192-cbc	Optional	MAY use with additional layer of security such as SSL / TLS	192 CBC plaintext Attack
	AES192- GCM http://www.w3.org/2009/xmlenc11#aes192-gcm	Optional	MAY	*192*
	AES256- GCM http://www.w3.org/2009/xmlenc11#aes256-gcm	Optional	MAY	*256*
Key Derivation	ConcatKDF http://www.w3.org/2009/xmlenc11#ConcatKDF	Required	SHOULD	*128 with use of SHA 1* *256 with use of SHA 256*

Type	Algorithm (URI)	Implementation	Recommendation on Applying	Level of Security
	PBKDF 2 http://www.w3.org/2009/xmlenc11#pbkdf2	Optional	MAY	*128 with use of SHA 1* *256 with use of SHA 256*
Key Transport	RSA - OAEP (including MGF1 with SHA 1) http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p	Required	SHOULD (k=2048)	80(k=1024) 112(k=2048) 128+ (k=3072+)
	RSA - OAEP http://www.w3.org/2009/xmlenc11#rsa-oaep	Optional	MAY	80(k=1024) 112(k=2048) 128+ (k=3072+)
	RSA -v1.5 http://www.w3.org/2001/04/xmlenc#rsa-1_5	Optional	MUST NOT	80+ (k=1024+) PKCS #1.5 attacks
Key Agreement	Elliptic Curve Diffie-Hellman (Ephemeral-Static mode) http://www.w3.org/2009/xmlenc11#ECDH-ES	Required	SHOULD	128
	Diffie-Hellman Key Agreement (Ephemeral-Static mode) with Legacy Key Derivation Function http://www.w3.org/2001/04/xmlenc#dh	Optional	MAY	80+ [a]
	Diffie-Hellman Key Agreement (Ephemeral-Static mode) with explicit Key Derivation Functions http://www.w3.org/2009/xmlenc11#dh-es	Optional	MAY	128+[b]

Type	Algorithm (URI)	Implementation	Recommendation on Applying	Level of Security
Symmetric Key Wrap	TRIPLEDES KeyWrap http://www.w3.org/2001/04/xmlenc#kw-tripledes	Required	MAY	112 Disallowed to apply and legacy use only beyond 2031
	AES -128 KeyWrap http://www.w3.org/2001/04/xmlenc#kw-aes128	Required	SHOULD	128
	AES -256 KeyWrap http://www.w3.org/2001/04/xmlenc#kw-aes256	Required	MAY	256
	AES -192 KeyWrap http://www.w3.org/2001/04/xmlenc#kw-aes192	Optional	MAY	192
Encoding	base64	Required	MUST	None
Transforms	See XMLDSig			

[a]The level of security is dependent on the choice of encryption algorithm and digest algorithm.

[b] ConcatKDF and PBKDF may be used for the algorithm. The level of security is derived from ConcatKDF and PBKDF .

Appendix D W3C Best Practice Summary

The W3C XML Security Working Group has collected the best practices for implementers and users of the XML Signature specification. Most of these best practices are related to improving security and mitigating attacks, yet others are for best practices in the practical use of XML Signature. The following is the summary of W3C Best practices for XML Signature. For more details and examples, please refer to http://www.w3.org/TR/xmldsig-bestpractices/ [31]

Best Practices for Implementers

- Mitigate denial of service attacks by executing potentially dangerous operations only after successfully authenticating the signature.
- Establish trust in the verification/validation key.
- Consider avoiding XSLT Transforms.
- When XSLT is required disallow the use of user-defined extensions.
- Try to avoid or limit XPath transforms.
- Avoid using the “descendant”, “descendant-or-self”, “following-sibling”, and “following” axes when using streaming XPaths .
- Try to avoid or limit ds:RetrievalMethod support with ds:KeyInfo.
- Control external references.
- Limit number of ds:Reference transforms allowed.
- Offer interfaces for application to learn what was signed.
- Do not re-encode certificates, use DER when possible with the X509Certificate element.

Best Practices for Applications

- Enable verifier to automate “see what is signed” functionality.
- When applying XML Signatures using XPath it is recommended to always actively verify that the signature protects the intended elements and not more or less.
- When checking a reference URI , don’t just check the name of the element.
- Unless impractical, sign all parts of the document.
- Use a nonce in combination with signing time.
- Do not rely on application logic to prevent replay attacks since applications may change.
- Nonce and signing time must be signature protected.

- Use Timestamp tokens issued by Timestamp authorities for long lived signatures.
- Long lived signatures should include a xsd:dateTime field to indicate the time of signing just as a handwritten signature does.
- When creating an enveloping signature over XML without namespace information, take steps to avoid having that content inherit the XML Signature namespace.
- Prefer the XPath Filter 2 Transform to the XPath Filter Transform if possible.
- Do not transmit unparsed external entity references.
- Do not rely on a validating processor on the consumer’s end.
- Avoid destructive validation before signature validation.
- When using an HMAC , set the HMAC Output Length to one half the number of bits in the hash size.
- When encrypting and signing use distinct keys.

Appendix E NIST Algorithm analysis

NIST Special Publication 800-57: Recommendation for Key Management – Part 1: General (Revision 3)[15] provides general guidance and best practices for the management of cryptographic keying material. The security strength of the cryptographic algorithms and the security lifetime are reported in this special publication. The security strength data can be used to acquire a cryptographic systems with appropriate algorithm and key sizes to provide adequate protection for the expected lifetime of the system or data. The recommendations on cryptographic algorithms in Section 3.2.2 - Consolidated Guidelines for Cryptographic Algorithms and Appendix C - Complete Guide for All XMLDSig and XMLEnc Defined Algorithms are based on the algorithm analysis results provided in the publication.

In this section, the original security strength data and security lifetime data reported in NIST SP800-57 are shown for comparison purpose. Please refer to NIST SP800-57 for more details.

Table 22 shows the security strength of various encryption algorithms.

Table 22 - Comparable Security Strengh of Various Encryption Algorithms

Bits of security	Symmetric key algorithms	FFC (e.g., DSA, D-H)	IFC (e.g., RSA)	ECC (e.g., ECDSA)
80	2TDEA	L = 1024 N = 160	k = 1024	f = 160-223
112	3TDEA	L = 2048 N = 224	k = 2048	f = 224-255
128	AES -128	L = 3072 N = 256	k = 3072	f = 256-383
192	AES -192	L = 7680 N = 384	k = 7680	f = 384-511
256	AES -256	L = 15360 N = 512	k = 15360	f = 512+

Table 23 shows the security strength of various hash algorithms.

Table 23 - Comparable Security Strength of Various Hash Algorithms

Security Strength	Digital Signatures and hash-only applications	HMAC	Key Derivation Functions	Random Number Generation
80	SHA -1, SHA -224, SHA -512/224, SHA -256, SHA -512/256, SHA -384, SHA -512	SHA -1, SHA -512/224, SHA -224, SHA -256, SHA -512/256, SHA -384, SHA -512	SHA -1, SHA -224, SHA -512/224, SHA -256, SHA -512/256, SHA -384, SHA -512	SHA -1, SHA -224, SHA -512/224, SHA -256, SHA -512-/256, SHA -384, SHA -512
112	SHA -224, SHA -512/224, SHA -256, SHA -512/256, SHA -384, SHA -512	SHA -1, SHA -224, SHA -512/224, SHA -256, SHA -512/256, SHA -384, SHA -512	SHA -1, SHA -224, SHA -512/224, SHA -256, SHA -512/256, SHA -384, SHA -512	SHA -1, SHA -224, SHA -512/224, SHA -256, SHA -512/256, SHA -384, SHA -512
128	SHA -256, SHA -512/256, SHA -384, SHA -512	SHA -1, SHA -224, SHA -512/224, SHA -256, SHA -512/256, SHA -384, SHA -512	SHA -1, SHA -224, SHA -512/224, SHA -256, SHA -512/256, SHA -384, SHA -512	SHA -1, SHA -224, SHA -512/224, SHA -256, SHA -512/256, SHA -384, SHA -512
192	SHA -384, SHA -512	SHA -224, SHA -512/224, SHA -256, SHA -512/256, SHA -384, SHA -512	SHA -224, SHA -512/224, SHA -256, SHA -512/256, SHA -384, SHA -512	SHA -224, SHA -512/224, SHA -256, SHA -512/256, SHA -384, SHA -512
256	SHA -512	SHA -256, SHA -512/256, SHA -384, SHA -512	SHA -256, SHA -512/256, SHA -384, SHA -512	SHA -256, SHA -512/256, SHA -384, SHA -512

Table 24 associates the security strength to the recommended security time frame for both applying and processing.

Table 24 - Security-strength time frames

Security Strength	Usage	2011 through 2013	2014 through 2030	2031 and Beyond
80	Applying	Deprecated	Disallowed	
	Processing	Legacy use		
112	Applying	Acceptable	Acceptable	Disallowed
	Processing	Acceptable	Acceptable	Legacy use
128	Applying/Processing	Acceptable	Acceptable	Acceptable
192	Applying/Processing	Acceptable	Acceptable	Acceptable

Security Strength	Usage	2011 through 2013	2014 through 2030	2031 and Beyond
256	Applying/Processing	Acceptable	Acceptable	Acceptable

Appendix F Glossary

This appendix lists all the acronyms and abbreviations referenced in this encoding specification.

2TDEA	double-length key Triple Data Encryption Algorithm
3TDEA	triple-length key Triple Data Encryption Algorithm
AES	Advanced Encryption Standard
ANSI	American National Standards Institute
BI	Binding Information
CAS	Central Authentication Service
CB	Cryptographic Binding
CBC	Cipher-Block Chaining
CDR	Content Discovery and Retrieval
CIO	Chief Information Officer
CMS	Cryptographic Message Syntax
CNSSI	Committee on National Security Systems Instruction
ConcatKDF	Concatenation Key Derivation Function
CONOPS	Concept of Operations
CSP	Critical Security Parameter
CVE	Controlled Vocabulary Enumeration
DER	Distinguished Encoding Rules
DNI	Director of National Intelligence
DOD	Department of Defense
DOS	Denial of Service
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPS	Federal Information Processing Standards
GCM	Galois Counter Mode

GMAC	Galois Message Authentication Code
HMAC	Hash-based Message Authentication Code
HTTP	Hypertext Transfer Protocol
IA	Information Assurance
IC	Intelligence Community
IC CIO	Intelligence Community Chief Information Officer
IC EA	Intelligence Community Enterprise Architecture
IC ESB	Intelligence Community Enterprise Standards Baseline
IC ITE	IC Information Technology Enterprise
ICD	Intelligence Community Directive
ICPG	Intelligence Community Program Guidance
ICS	Intelligence Community Standard
ISOO	Information Security Oversight Office
IT	Information Technology
ITL	Information Technology Laboratory
JSON	JavaScript Object Notation
JWE	JSON Web Encryption
JWS	JSON Web Signature
JWT	JSON Web Token
MAC	Multi Audience Collection
MGF1	Mask Generation Function based on a hash function
NIST	National Institute of Standards and Technology
NSA	National Security Agency
OAEP	Optimal Asymmetric Encryption Padding Scheme
OASIS	Organization for the Advancement of Structured Information Standards
OCIO	Office of the Intelligence Community Chief Information Officer

ODNI	Office of the Director of National Intelligence
OS	Operating System
PAOS	Reverse SOAP
PBKDF	Password-Based Key Derivation Function
PKCS	Public-Key Cryptography Standards
RBG	Random Bit Generator
REST	Representational State Transfer
RSA	RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman, who first publicly described a public-key cryptosystems algorithm in 1977
RR-ID	REST Security Encoding Specification for End-to-End Identity Propagation
SAML	Security Assertion Markup Language
SFTP	Secure File Transfer Program
SHA	Secure Hash Algorithm
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SP	Special Publication
SSL	Secure Sockets Layer
SSO	Single Sign-On
STS	Security Token Service
TLS	Transport Layer Security
TRIPLEDES	Triple Data Encryption Algorithm
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WS	Web Service
WSSWG	Web Services Security Working Group

X.509	ITU-T standard for public key infrastructures
XACML	Extensible Access Control Markup Language
XML	Extensible Markup Language
XMLEnc	XML Encryption Syntax and Processing
XMLSig	XML-Signature Syntax and Processing
XPath	XML Path Language
XSL	Extensible Stylesheet Language
XSLT	XSL Transformations

Appendix G Bibliography

Bibliography

[1] ATOM.XML
Intelligence Community / Department of Defense Content Discovery & Retrieval Integrated Project Team. *Atom Data Encoding Specification for Content Discovery and Retrieval Result Sets*.
Available online Intelink-U at: http://purl.org/IC/Standards/ATOM
Available online at: http://purl.org/IC/Standards/public

[2] FIPS 140-2
National Institute of Standards and Technology. *Security Requirements for Cryptographic Module*. . May 2001.
Available online at: http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

[3] FIPS 186-4
National Institute of Standards and Technology. *Digital Signature Standard (DSS)*. FIPS 186-4. July 2013.
Available online at: http://csrc.nist.gov/publications/fips/fips186-4/fips_186-4.pdf

[4] IC ITE INC1 IMPL
Office of the Director of National Intelligence. *Intelligence Community Information Technology Enterprise (IC ITE) Increment 1 Implementation Plan*. July 2012.
Available online Intelink-TS at: http://go.ic.gov/HvBHBmY

[5] IC-TDF.XML
Office of the Director of National Intelligence. *XML Data Encoding Specification for Trusted Data Format (TDF.XML)*.
Available online Intelink-U at: http://purl.org/IC/Standards/TDF
Available online at: http://purl.org/IC/Standards/public

[6] IC-WSS-HLG.XML
Office of the Director of National Intelligence. *High Level Guidance for Web Service Security*.
Available online IntelLinkU at: http://purl.org/IC/Standards/TDF
Available online at: http://purl.org/IC/Standards/public

[7] ICD 500
Office of the Director of National Intelligence. *Director of National Intelligence Chief Information Officer*. Intelligence Community Directive 500. 7 August 2008.
Available online Intelink-TS at: http://go.ic.gov/enm8L9x
Available online at: http://www.dni.gov/files/documents/ICD/ICD_500.pdf

[8] ICPG 500.1
Assistant Director of National Intelligence for. *Digital Identity*. Intelligence Community Policy Guidance 500.1. 7 May 2010.
Available online Intelink-TS at: http://go.ic.gov/3rfgL6D

[9] ICPG 500.2
Assistant Director of National Intelligence for Policy and Strategy. *Attribute-Based Authorization and Access Management*. Intelligence Community Policy Guidance 500.2. 23 November 2010.
Available online Intelink-TS at: http://go.ic.gov/ha2FxyZ
Available online at: http://www.dni.gov/files/documents/ICPG/icpg_500_2.pdf

[10] ICPG 710.1
Assistant Director of National Intelligence for . *Application of Dissemination Controls: Originator Control*. Intelligence Community Policy Guidance 710.1. 25 July 2012.
Available online Intelink-TS at: http://go.ic.gov/yAqVQ0H

[11] ICS 500-20
Director of National Intelligence Chief Information Officer. *Intelligence Community Enterprise Standards Compliance*. Intelligence Community Standard 500-20. 16 December 2010.
Available online Intelink-TS at: http://go.ic.gov/QUDIJkZ
Available online Intelink-U at: https://intelshare.intelink.gov/sites/odni/cio/ea/library/Data%20Specifications/500-21/500_20_signed_16DEC2010.pdf

[12] IETF-RFC 2119
Internet Engineering Task Force. *Key words for use in RFCs to Indicate Requirement Levels*. March 1997.
Available online at: http://tools.ietf.org/html/rfc2119

[13] IETF-RFC 4287
M. Nottingham, R. Sayre. *The Atom Syndication Format*. December 2005.
Available online at: http://www.ietf.org/rfc/rfc4287.txt

[14] NIST 800-38D
National Institute of Standards and Technology. *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. . Nov 2007.
Available online at: http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf

[15] NIST 800-57
National Institute of Standards and Technology. *Recommendation for Key Management – Part 1: General* . Revision 3. July 2012.
Available online at: http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf

[16] NIST Type98 guide
National Institute of Standards and Technology. *ANSI/NIST ITL 2011 Type 98 Best Practice Implementation Guidance* . Revision 1.3. June 2011.
Available online at: http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf

[17] NSA CryptoBinding CONOPS
National Security Agency. *Cryptographic Binding CONOPS*. 2. Sept 2008.

Available online at: https://inteldocs.intelink.gov/inteldocs/proxy/alfresco/api/node/content/workspace/SpacesStore/407de0c1-9047-4c6b-9a34-fd021eab4222/Vol2+Cryptographic+Binding+CONOPS+V1.2+(16-Dec-08).doc

[18] NSA Rest Guide
National Security Agency. *Guidelines for Implementation of REST*. Mar 2011.
Available online at: http://www.nsa.gov/ia/_files/support/guidelines_implementation_rest.pdf [http://www.nsa.gov/ia/_files/support/guidelines_implementation_rest.pdf]

[19] NSA Rest Whitepaper
National Security Agency. *REST White Paer*. Mar 2009.
Available online at: https://inteldocs.intelink.gov/inteldocs/proxy/alfresco/api/node/content/workspace/SpacesStore/d8a555d4-f41c-433b-8cd6-2b26a3e28ad6/REST+Whitepaper+FINAL+16March2009.pdf

[20] REST
R Fielding, R Taylor. *Principled Design of the Modern Web Architecture*. ACM Transactions on Internet Technology. Vol 2, No. 2, May 2002, pages 115-150.
Available online at: http://www.ics.uci.edu/~taylor/documents/2002-REST-TOIT.pdf [http://www.ics.uci.edu/~taylor/documents/2002-REST-TOIT.pdf]

[21] RR-ID.XML
Office of the Director of National Intelligence. *REST Service Encoding Specification for End-to-End Identity Propagation (RR-ID.XML)*.
Available online Intelink-U at: http://purl.org/IC/Standards/RR-ID [http://purl.org/IC/Standards/PUBS]
Available online at: http://purl.org/IC/Standards/public

[22] SAML 2.0
Organization for the Advancement of Structured Information Standards. *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. 27 March 2008.
Available online at: http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

[23] SAML 2.0 Web Browser SSO Profile
Organization for the Advancement of Structured Information Standards. *Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0*. March 15, 2005.
Available online at: http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf

[24] SAML 2.0 Technical Overview
Organization for the Advancement of Structured Information Standards. *Security Assertion Markup Language (SAML) V2.0 Technical Overview*. March 20, 2008.
Available online at: https://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf

[25] SOAP
World Wide Web Consortium (W3C). *SOAP Version 1.2 Part 1: Messaging Framework*. W3C Recommendation 27 April 2007.
Available online at: http://www.w3.org/TR/soap12-part1/

[26] WS-Security
The Organization for the Advancement of Structured Information Standards (OASIS). *OASIS Web Services Security: SOAP Message Security*. V1.1.1 May 2012.
Available online at: http://www.w3.org/TR/ws-addr-soap/

[27] XACML
The Organization for the Advancement of Structured Information Standards (OASIS). *OASIS Extensible Access Control Markup Language (XACML) v. 3.0*. Jan 2013.
Available online at: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.pdf

[28] GeoXACML
Open Geospatial Consortium. *Geospatial Extensible Access Control Markup Language (GeoXACML) Version 1* . May 2011.
Available online at: http://www.opengeospatial.org/standards/geoxacml

[29] XML Encryption
World Wide Web Consortium (W3C). *W3C XML Encryption Syntax and Processing, Version 1.1*. April 2013.
Available online at: http://www.w3.org/TR/2013/REC-xmlenc-core1-20130411/

[30] XML Signature
World Wide Web Consortium (W3C). *W3C XML Signature Syntax and Processing, Version 1.1*. April 2013.
Available online at: http://www.w3.org/TR/2013/REC-xmldsig-core1-20130411/

[31] XML Signature Guide
World Wide Web Consortium (W3C). *XML Signature Best Practices W3C Working Group Note* . April 2013.
Available online at: http://www.w3.org/TR/xmldsig-bestpractices/

Appendix H Points of Contact

The Intelligence Community Chief Information Officer (IC CIO) facilitates one or more collaboration and coordination forums charged with the adoption, modification, development, and governance of IC technical specifications of common concern. This technical specification was produced by the IC CIO and coordinated with these forums, approved by the IC CIO or a designated representative, and made available at DNI -sponsored web sites. Direct all inquiries about this IC technical specification to the IC CIO, an IC technical specification collaboration and coordination forum, or IC element representatives involved in those forums.

Public Website: http://purl.org/ic/standards/public

E-mail: ic-standards-support@intelink.gov [mailto:ic-standards-support@intelink.gov].

Appendix I IC CIO Approval Memo

An Office of the Intelligence Community Chief Information Officer (OCIO) Approval Memo should accompany this enterprise technical data specification bearing the signature of the Intelligence Community Chief Information Officer (IC CIO) or an IC CIO -designated official(s). If an OCIO Approval Memo is not accompanying this specification's version release package, then refer back to the authoritative web location(s) for this specification to see if a more complete package or a specification update is available.

Specification artifacts display a date representing the last time a version's artifacts as a whole were modified. This date most often represents the conclusion of the IC Element collaboration and coordination process. Once the IC Element coordination process is complete, the specification goes through an internal OCIO staffing and coordination process leading to signature of the OCIO Approval Memo. The signature date of the OCIO Approval Memo will be later than the last modified date shown on the specification artifacts by an indeterminable time period.

Upon signature of the OCIO Approval Memo, IC Elements may begin to use this specification version in order to address mission and business objectives. However, it is critical for IC Elements, prior to disseminating information encoded with this new specification version, to ensure that key enterprise services and consumers are prepared to accept this information. IC Elements should work with enterprise service providers and consumers to orchestrate an orderly implementation transition to this specification version in concert with mandatory and retirement usage decisions captured in the IC Enterprise Standards Baseline as defined in Intelligence Community Standard (ICS) 500-20.[11]