Long-term Archive And Notary	T. Kunz
Services (LTANS)	Fraunhofer Institute for Secure
Internet-Draft	Information Technology
Intended status: Standards Track	S. Okunick
Expires: December 17, 2009	pawisda systems GmbH
	U. Pordesch
	Fraunhofer Gesellschaft
	June 15, 2009

TOC

Data Structure for the Security Suitability of Cryptographic Algorithms (DSSC)
draft-ietf-ltans-dssc-09.txt

Status of this Memo

Copyright Notice

Abstract

Since cryptographic algorithms can become weak over the years, it is necessary to evaluate their security suitability. When signing or verifying data, or when encrypting or decrypting data, these evaluations must be considered. This document specifies a data structure that enables an automated analysis of the security suitability of a given cryptographic algorithm at a given point of time which may be in the past, at the present time or in the future.

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.).

Table of Contents

---

1. Introduction

TOC

1.1. Motivation

TOC

Digital signatures can provide data integrity and authentication. They are based on cryptographic algorithms that are required to have certain security properties. For example, hash algorithms must be resistant to collisions and in case of public key algorithms computation of the private key that corresponds to a given public key must be infeasible. If algorithms lack the required properties, signatures could be forged, unless they are protected by a strong cryptographic algorithm. Cryptographic algorithms that are used in signatures shall be selected to resist such attacks during their period of use. For signature keys included in public key certificates, it is the validity period of the certificate. Cryptographic algorithms that are used for encryption shall resist during the time during which it is planned to keep the information confidential.
Only very few algorithms satisfy the security requirements. Besides, because of the increasing performance of computers and progresses in cryptography, algorithms or their parameters become insecure over the years. The hash algorithm MD5, for example, is unsuitable today for many purposes. A digital signature using a "weak" algorithm has no probative value, unless the "weak" algorithm has been protected by a strong algorithm before the time it was considered to be weak. Many kinds of digital signed data, including signed documents, time stamps,

certificates, and revocation lists, are affected, in particular in the case of long-term archiving. Over long periods of time, it is assumed that the algorithms used in signatures become insecure.
For this reason, it is important to periodically evaluate an algorithm's fitness and to consider the results of these evaluations when creating and verifying signatures, or when maintaining the validity of signatures made in the past. One result is a projected validity period for the algorithm, i.e., a prediction of the period of time during which the algorithm is fit for use. This prediction can help to detect whether a weak algorithm is used in a signature and whether that signature has been properly protected in due time by another signature made using an algorithm that is suitable at the present point of time. Algorithm evaluations are made by expert committees. In Germany the Federal Network Agency annually publishes evaluations of cryptographic algorithms [BNetzAg.2008] (Federal Network Agency for Electricity, Gas, Telecommunications, Post and Railway, “Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Übersicht über geeignete Algorithmen),” December 2007.). Examples of other European and international evaluations are [ETSI-TS102176-1-2005] (European Telecommunication Standards Institute (ETSI), “Electronic Signatures and Infrastructures (ESI); "Algorithms and Parameters for Secure Electronic Signatures; Part 1: Hash functions and asymmetric algorithms",” November 2007.) and [NIST.800-57-Part1.2006] (National Institute of Standards and Technology, “Recommendation for Key Management – Part 1: General (Revised),” May 2006.).
These evaluations are published in documents intended to be read by humans. Therefore it is necessary to define a data structure that expresses the content of the evaluations to enable automated processing. This standardized data structure can be used for publication and can be interpreted by signature generation and verification tools. Algorithm evaluations are pooled in a security suitability policy. In this document a data structure for a security suitability policy is specified. This document does not attempt to catalog the security properties of cryptographic algorithms.

1.2. Terminology

TOC

Algorithm: A cryptographic algorithm, i.e. a public key or hash algorithm. For public key algorithms, this is the algorithm with its parameters, if any. Furthermore, the term 'algorithm' is used for combinations of public key and hash algorithms, and actually padding functions (e.g. the signature algorithm SHA-1 with RSA).

Operator: Instance which uses and interprets a policy, e.g. a signature verification component.

Policy:
An abbreviation for security suitability policy.

Publisher: Instance that publishes the policy containing the evaluation of algorithms.

Security suitability policy: The evaluation of cryptographic algorithms with regard to their security in a specific application area, e.g. signing or verifying data. The evaluation is published in an electronic format.

Suitable algorithm: An algorithm which is evaluated against a policy and determined to be valid, i.e. resistant against attacks, at a particular point of time.

1.3. Use Cases

TOC

In the following some use cases for a security suitability policy are presented.

Long-term archiving: The most important use case is long-term archiving of signed data. Algorithms or their parameters become insecure over long time periods. Therefore signatures of archived data and timestamps have to be periodically renewed. A policy provides information about suitable and threatened algorithms. Additionally the policy assists in verifying archived as well as re-signed documents.

Services: Services may provide information about cryptographic algorithms. On the basis of a policy a service is able to provide the date when an algorithm became insecure or presumably will become insecure or to provide all algorithms which are presently valid. Verification tools or long-term archiving systems can request such services and therefore do not need to deal with the algorithm security by themselves. Long-term Archive Services (LTA) as defined in [RFC4810] (Wallace, C., Pordesch, U., and R. Brandner, “Long-Term Archive Service Requirements,” March 2007.) may use the policy for signature renewal.

Signing and verifying: When signing documents, or certificates, it must be assured that the algorithms used for signing or verifying are suitable. Accordingly, when verifying CMS [RFC3852] (Housley, R., “Cryptographic Message Syntax (CMS),” July 2004.) or XML signatures [RFC3275] (Eastlake, D., Reagle, J., and D. Solo, “(Extensible Markup Language) XML-Signature Syntax and Processing,” March 2002.) [ETSI-TS101903] (European

Telecommunication Standards Institute (ETSI), “XML Advanced Electronic Signatures (XAdES),” March 2006.), not only the validity of the certificates may be checked but also the validity of the algorithms.

Re-encryption: A security suitability policy can also be used to decide if encrypted documents must be re-encrypted because the encryption algorithm is no longer secure.

---

2. Requirements and Assumptions

TOC

Section 2.1 (Requirements) describes general requirements for a data structure containing the security suitability of algorithms. In Section 2.2 (Assumptions) assumptions are specified concerning both the design and the usage of the data structure.
A policy contains a list of algorithms that have been evaluated by a publisher. An algorithm evaluation is described by its identifier, security constraints and validity period. By these constraints the requirements for algorithm properties must be defined, e.g. a public key algorithm is evaluated on the basis of its parameters.

2.1. Requirements

TOC

Automatic interpretation: The data structure of the policy must allow automated evaluation of the security suitability of an algorithm.

Flexibility: The data structure must be flexible enough to support new algorithms. Future policy publications may include evaluations of algorithms that are currently unknown. It must be possible to add new algorithms with the corresponding security constraints in the data structure. Additionally the data structure must be independent of the intended use, e.g., encryption, signing, verifying, and signature renewing. Thus, the data struture is usable in every use case.

Source authentication: Policies may be published by different institutions, e.g. on national or EU level, whereas one policy needs not to be in agreement with the other one. Furthermore organizations may undertake their own evaluations for internal purposes. For this reason a policy must be attributable to its publisher.

Integrity and authenticity:

> It must be possible to assure the integrity and authenticity of a published security suitability policy. Additionally the date of issue must be identifiable.

2.2. Assumptions

TOC

It is assumed that a policy contains the evaluations of all currently known algorithms, including the expired ones.
An algorithm is suitable at a time of interest if it is contained in the current policy and the time of interest is within the validity period. Additionally, if the algorithm has any parameters, these parameters must meet the requirements defined in the security constraints.
If an algorithm appears in a policy for the first time, it may be assumed that the algorithm has already been suitable in the past. Generally, algorithms are used in practice prior to evaluation.
To avoid inconsistencies, multiple instances of the same algorithm are prohibited. The publisher must take care about preventing conflicts within a policy.
Assertions made in the policy are suitable at least until the next policy is published.
Publishers may extend the lifetime of an algorithm prior to reaching the end of the algorithm's validity period by publishing a revised policy. Publishers should not resurrect algorithms that are expired at the time a revised policy is published.

---

3. Data Structures

TOC

This section describes the syntax of a security suitability policy defined as an XML schema. ASN.1 modules are defined in Appendix C (ASN.1 Module in 1988 Syntax (informative)) and Appendix D (ASN.1 Module in 1997 Syntax (normative)). The schema uses the following namespace:

> http://www.sit.fraunhofer.de/dssc

Within this document, the prefix "dssc" is used for this namespace. The schema starts with the following schema definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:dssc="http://www.sit.fraunhofer.de/dssc"
           xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
           targetNamespace="http://www.sit.fraunhofer.de/dssc"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
<xs:import namespace="http://www.w3.org/XML/1998/namespace"
           schemaLocation="http://www.w3.org/2001/xml.xsd"/>
<xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
           schemaLocation="xmldsig-core-schema.xsd"/>
```

3.1. SecuritySuitabilityPolicy

TOC

The SecuritySuitabilityPolicy element is the root element of a policy. It has an optional id attribute which must be used as a reference when signing the policy (Section 3.14 (Signature)). The element is defined by the following schema:

```
<xs:element name="SecuritySuitabilityPolicy"
            type="dssc:SecuritySuitabilityPolicyType"/>
<xs:complexType name="SecuritySuitabilityPolicyType">
  <xs:sequence>
    <xs:element ref="dssc:PolicyName"/>
    <xs:element ref="dssc:Publisher"/>
    <xs:element name="PolicyIssueDate" type="xs:dateTime"/>
    <xs:element name="NextUpdate" type="xs:dateTime" minOccurs="0"/>
    <xs:element name="Usage" type="xs:string" minOccurs="0"/>
    <xs:element ref="dssc:Algorithm" maxOccurs="unbounded"/>
    <xs:element ref="ds:Signature" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="version" type="xs:string" default="1"/>
  <xs:attribute name="id" type="xs:ID"/>
</xs:complexType>
```

TOC

3.2. PolicyName

The PolicyName element consists of an arbitrary name of the policy and an optional Uniform Resource Identifier (URI).

```
<xs:element name="PolicyName" type="dssc:PolicyNameType"/>
<xs:complexType name="PolicyNameType">
  <xs:sequence>
    <xs:element ref="dssc:Name"/>
    <xs:element ref="dssc:URI" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="Name" type="xs:string"/>
<xs:element name="URI" type="xs:anyURI"/>
```

3.3. Publisher

TOC

The Publisher element contains information about the publisher of the policy. It is composed of the name, e.g. name of institution, an optional address, and an optional URI.

```
<xs:element name="Publisher" type="dssc:PublisherType"/>
<xs:complexType name="PublisherType">
  <xs:sequence>
    <xs:element ref="dssc:Name"/>
    <xs:element ref="dssc:Address" minOccurs="0"/>
    <xs:element ref="dssc:URI" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

3.4. Address

TOC

The Address element consists of the street, the locality, the optional state or province, the postal code, and the country.

```
<xs:element name="Address" type="dssc:AddressType"/>
<xs:complexType name="AddressType">
  <xs:sequence>
    <xs:element name="Street" type="xs:string"/>
    <xs:element name="Locality" type="xs:string"/>
    <xs:element name="StateOrProvince" type="xs:string" minOccurs="0"/>
    <xs:element name="PostalCode" type="xs:string"/>
    <xs:element name="Country" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

3.5. PolicyIssueDate

TOC

The PolicyIssueDate element indicates the point of time when the policy was issued.

3.6. NextUpdate

TOC

The optional NextUpdate element may be used to indicate when the next policy will be issued.

3.7. Usage

TOC

The optional Usage element determines the intended use of the policy (e.g. certificate validation, signing and verifying documents).

3.8. Algorithm

TOC

A security suitability policy must contain at least one Algorithm element. An algorithm is identified by an AlgorithmIdentifier element. Additionally the Algorithm element contains all evaluations of the specific cryptographic algorithm. More than one evaluation may be necessary if the evaluation depends on the parameter constraints. The Algorithm element is defined by the following schema:

```
<xs:element name="Algorithm" type="dssc:AlgorithmType"/>
<xs:complexType name="AlgorithmType">
  <xs:sequence>
    <xs:element ref="dssc:AlgorithmIdentifier"/>
    <xs:element ref="dssc:Evaluation" maxOccurs="unbounded"/>
    <xs:element ref="dssc:Information" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

3.9. AlgorithmIdentifier

TOC

The AlgorithmIdentifier element is used to identify a cryptographic algorithm. It consists of the algorithm name, at least one object identifer, and optional URIs. The element is defined as follows:

```
<xs:element name="AlgorithmIdentifier"
            type="dssc:AlgorithmIdentifierType"/>
<xs:complexType name="AlgorithmIdentifierType">
  <xs:sequence>
    <xs:element ref="dssc:Name"/>
    <xs:element name="ObjectIdentifier" type="xs:string"
                maxOccurs="unbounded"/>
    <xs:element ref="dssc:URI" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

3.10. Evaluation

TOC

The evaluation element contains the evaluation of one cryptographic algorithm in dependence of its parameter contraints. E.g. the suitability of the RSA algorithm depends on the modulus length (RSA with a modulus length of 1024 may have another suitability period as RSA with a modulus length of 2048). Current hash algorithms like SHA-1 or RIPEMD-160 do not have any parameters. Therefore the Parameter element is optional. The suitability of the algorithm is expressed by a validity period which is defined by the Validity element.

```
<xs:element name="Evaluation" type="dssc:EvaluationType"/>
<xs:complexType name="EvaluationType">
  <xs:sequence>
    <xs:element ref="dssc:Parameter" minOccurs="0"
                                     maxOccurs="unbounded"/>
    <xs:element ref="dssc:Validity"/>
  </xs:sequence>
</xs:complexType>
```

3.11. Parameter

TOC

The Parameter element is used to express constraints on algorithm specific parameters like the "moduluslength" parameter in case of RSA. The Parameter element has a name attribute which holds the name of the parameter (e.g. "moduluslength" for RSA [RFC3447] (Jonsson, J. and B. Kaliski, “Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1,” February 2003.). Besides a better readability of the policy, the attribute may be used by implementations for output messages. In Section 4 (Definition of Parameters) the parameter names of currently known signature algorithms are defined. For the actual parameter, an exact value or a range of values may be defined. These constraints are expressed by the following elements:

> **Exact:** The Exact element specifies the exact value of the parameter.
>
> **Min:** The Min element defines the minimum value of the parameter. That means, also all other values greater than the given one meet the requirements.
>
> **Max:** The Max element defines the maximum value the parameter may take.
>
> **Range:** The Range element is used to define a range of values, consisting of a minimum and a maximum value. The parameter may have any value within the defined range, including the minimum and maximum values.

For one algorithm it is recommended not to mix these elements in order to avoid inconsistencies.
These constraints are sufficient for all current algorithms. If future algorithms will need constraints which cannot be expressed by the elements above, an arbitrary XML structure may be inserted which meets

the new constraints. For this reason, the Parameter element contains an "any" element. The schema for the Parameter element is as follows:

```
<xs:element name="Parameter" type="dssc:ParameterType"/>
<xs:complexType name="ParameterType">
  <xs:choice>
    <xs:element name="Exact" type="xs:string"/>
    <xs:element ref="dssc:Min"/>
    <xs:element ref="dssc:Max"/>
    <xs:element name="Range">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="dssc:Min"/>
          <xs:element ref="dssc:Max"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:any namespace="##other"/>
  </xs:choice>
  <xs:attribute name="name" type="xs:string" use="required"/>
</xs:complexType>
<xs:element name="Min" type="xs:string"/>
<xs:element name="Max" type="xs:string"/>
```

3.12. Validity

TOC

The Validity element is used to define the period of the (predicted) suitability of the algorithm. It is composed of an optional start date and an optional end date. Defining no end date means the algorithm has an open-end validity. Of course this may be restricted by a future policy which sets an end date for the algorithm. If the end of the validity period is in the past, the algorithm was suitable until that end date. The element is defined by the following schema:

```
<xs:element name="Validity" type="dssc:ValidityType"/>
<xs:complexType name="ValidityType">
  <xs:sequence>
    <xs:element name="Start" type="xs:date" minOccurs="0"/>
    <xs:element name="End" type="xs:date" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

3.13. Information

TOC

The Information element may be used to give additional textual information about the algorithm or the evaluation, e.g. references on algorithm specifications. The element is defined as follows:

```
<xs:element name="Information" type="dssc:InformationType"/>
<xs:complexType name="InformationType">
  <xs:sequence>
    <xs:element name="Text" maxOccurs="unbounded">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string">
            <xs:attribute name="lang"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

3.14. Signature

TOC

The optional Signature element may be used to guarantee the integrity and authenticity of the policy. It is an XML signature specified in [RFC3275] (Eastlake, D., Reagle, J., and D. Solo, "(Extensible Markup Language) XML-Signature Syntax and Processing," March 2002.). The signature must relate to the SecuritySuitabilityPolicy element. If the Signature element is set, the SecuritySuitabilityPolicy element must have the optional id attribute. This attribute must be used to reference the SecuritySuitabilityPolicy element within the Signature element. Since it is an enveloped signature, the signature must use the transformation algorithm identified by the following URI:

```
http://www.w3.org/2000/09/xmldsig#enveloped-signature
```

TOC

4. Definition of Parameters

This section defines the parameter names for the currently known public key algorithms. The signature algorithms RSA [RFC3447] (Jonsson, J. and B. Kaliski, “Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1,” February 2003.) and DSA [FIPS186-2] (National Institute of Standards and Technology, “Digital Signature Standard (DSS),” January 2000.) are generally used in conjunction with a one-way hash algorithm. Examples of such combined algorithms are SHA-256 with RSA and SHA-1 with DSA. The following parameters refer to the appropriate combined algorithms as well.

> The parameter of RSA should be named "moduluslength".
>
> The parameters for DSA should be "plength" and "qlength".

Publishers of policies must use the same parameter names, so that the correct interpretation is guaranteed.
For future algorithms, it may be necessary to update the information in this section. We suggest to handle this by the means of the IETF standards action (e.g. an updating RFC, which defines the parameter names of new algorithms).

---

5. Processing

TOC

Evaluation of an algorithm's security suitability is described in three parts: verification of the policy, determination of algorithm validity, and evaluation of algorithm parameters, if any.
In the following, a process is described

- to determine if an algorithm was suitable at a particular point of time
- and to determine until when an algorithm was or will be suitable.

5.1. Inputs

TOC

To determine the security suitability of an algorithm, the following information is required:

- Policy
- Current time

*Algorithm identifier and parameter constraints (if associated)

*Time of interest (optional). Providing no time of interest means determination of the validity end date of algorithm.

5.2. Verify policy

TOC

The signature on the policy SHOULD be verified and a certification path from the policy signer's certificate to a current trust anchor SHOULD be constructed and validated [RFC5280] (Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,” May 2008.). The algorithms used to verify the digital signature and validate the certification path MUST be suitable per the contents of the policy being verified. If signature verification fails, certification path validation fails or an unsuitable algorithm is required to perform these checks, then the policy MUST be rejected.
The nextUpdate time in the policy MUST be greater than the current time or absent. If the nextUpdate time is less than the current time, the policy MUST be rejected.

5.3. Algorithm evaluation

TOC

To determine the validity period of an algorithm, locate the Algorithm element in the policy that corresponds to the algorithm identifier provided as input. The Algorithm element is located by comparing the object identifier in the element to the object identifier included in the algorithm identifier provided as input.
If no matching Algorithm element is found, then the algorithm is unknown.
If the time of interest was provided as input, the validity of each Evaluation element MUST be checked in order to determine if the algorithm was suitable at the time of interest. For each Evaluation element,

*Confirm the Start time is less than the time of interest or absent. Discard the entry if the Start time is present and greater than the time of interest.

*Confirm the End time is greater than the time of interest or absent. Discard the entry if the End time is present and less than the time of interest.

If all Evaluation elements were rejected, the algorithm is not suitable according the policy.
Any entries not rejected will be used for the evaluation of the parameters, if any.

5.4. Evaluation of parameters

TOC

Any necessary parameters of the entries not rejected MUST be evaluated within the context of the type and usage of the algorithm. Details of parameter evaluation are defined on a per algorithm basis.
To evaluate the parameters, the Parameter elements of each Evaluation element that has not been rejected in the process described in Section 5.3 (Algorithm evaluation) must be checked. For each Parameter element,

- Confirm that the parameter was provided as input. Discard the Evaluation element if the parameter does not match to any of the parameters provided as input.

- If the Parameter element has an Exact element, confirm that the parameter value exactly complies with the according parameter provided as input. Discard the Evaluation element if the parameter value does not comply.

- If the Parameter element has a Min element, confirm that the parameter value is less than or equal to the according parameter provided as input. Discard the Evaluation element if the parameter value does not meet the constraint.

- If the Parameter element has a Max element, confirm that the parameter value is greater than or equal to the according parameter provided as input. Discard the Evaluation element if the parameter value does not meet the constraint.

- If the Parameter element has a Range element, confirm that the value of the according parameter provided as input is within the range. Discard the Evaluation element if the parameter value does not meet the constraint.

- If the Parameter has another constraint, confirm that the value of the according parameter provided as input meets this constraint. If it does not or if the constraint is unrecognized, discard the Evaluation element.

If all Evaluation elements were rejected, the algorithm is not suitable according the policy.
Any entries not rejected will be provided as output.

5.5. Output

TOC

If the algorithm is not in the policy, return an error "algorithm unknown".
If no time of interest was provided as input, return the maximum End time of the Evaluation elements that were not discarded. If at least one End time of these Evaluation elements is absent, return "algorithm has an indefinite end time".
Otherwise, if the algorithm is not suitable relative to the time of interest, return an error "algorithm unsuitable".
If the algorithm is suitable relative to the time of interest, return the Evaluation elements that were not discarded.

6. Security Considerations

TOC

The policy for algorithm's security suitability has great impact on the quality of the results of signature generation and verification operations. If an algorithm is incorrectly evaluated against a policy, signatures with a low probative force could be created or verification results could be incorrect. The following security considerations have been identified:

1. Publishers must ensure unauthorized manipulation of any security suitability is not possible prior to a policy being signed and published. There is no mechanism provided to revoke a policy after publication. Since the algorithm evaluations change infrequently, the lifespan of a policy should be carefully considered prior to publication.

2. Operators should only accept policies issued by a trusted publisher. It must not be possible to alter or replace a security suitability once accepted by the client.

3. Operators should periodically check to see if a new policy has been published to avoid using obsolete policy information. For publishers it is suggested not to omit the NextUpdate element in order to give operators a hint, when the next policy will be published.

4. When signing a policy, algorithms should be used which are suitable according this policy.

5. The processing rule described in Section 5 (Processing) is about one cryptographic algorithm independently of the use case. Depending upon the use case, an algorithm that is no more suitable at the time of interest, does not necessarily mean that the data structure where it is used is no more secure. For example, a signature has been made with an RSA signer's key of 1024 bits. This signature is time-stamped with a time-stamp token that uses an RSA key of 2048 bits, before an RSA key size of 1024 bits will be broken. The fact that the signature key of 1024 bits is no more suitable at the time of interest does not mean that the whole data structure is no more secure, if an RSA key size of 2048 bits is still suitable at the time of interest.

6. In addition to the key size considerations, other considerations must be applied, like whether a time-stamp token has been provided by a trusted authority. It means that the simple use of a suitability policy is not the single element to consider when evaluating the security of a complex data structure using several cryptographic algorithms.

7. Re-encrypting documents that were originally encrypted using an algorithm that is no more suitable, will not protect the semantics of the document, if the document has been intercepted. However, for documents stored in an encrypted form, re-encryption must be considered, unless the document has lost its original value.

---

7. IANA Considerations

TOC

This document has no actions for IANA. Section can be removed prior to publication as an RFC.

---

8. References

TOC

8.1. Normative References

TOC

[RFC2119]	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).

[RFC3275]	Eastlake, D., Reagle, J., and D. Solo, “(Extensible Markup Language) XML-Signature Syntax and Processing,” RFC 3275, March 2002 (TXT).
[RFC3852]	Housley, R., “Cryptographic Message Syntax (CMS),” RFC 3852, July 2004 (TXT).
[RFC4998]	Gondrom, T., Brandner, R., and U. Pordesch, “Evidence Record Syntax (ERS),” RFC 4998, August 2007 (TXT).
[RFC5280]	Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,” RFC 5280, May 2008 (TXT).

8.2. Informative References

TOC

[BNetzAg.2008]	Federal Network Agency for Electricity, Gas, Telecommunications, Post and Railway, “Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Übersicht über geeignete Algorithmen),” December 2007.
[ETSI-TS101903]	European Telecommunication Standards Institute (ETSI), “XML Advanced Electronic Signatures (XAdES),” ETSI TS 101 903 V1.3.2, March 2006.
[ETSI-TS102176-1-2005]	European Telecommunication Standards Institute (ETSI), “Electronic Signatures and Infrastructures (ESI); "Algorithms and Parameters for Secure Electronic Signatures; Part 1: Hash functions and asymmetric algorithms",” ETSI TS 102 176-1 V2.0.0, November 2007.
[FIPS186-2]	National Institute of Standards and Technology, “Digital Signature Standard (DSS),” FIPS PUB 186-2 with Change Notice, January 2000.
[NIST.800-57-Part1.2006]	National Institute of Standards and Technology, “Recommendation for Key Management – Part 1: General (Revised),” NIST 800-57 Part1, May 2006.
[RFC3447]	Jonsson, J. and B. Kaliski, “Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1,” RFC 3447, February 2003 (TXT).
[RFC4810]	Wallace, C., Pordesch, U., and R. Brandner, “Long-Term Archive Service Requirements,” RFC 4810, March 2007 (TXT).

TOC

Appendix A. DSSC and ERS

A.1. Verification of Evidence Records using DSSC (informative) TOC

This section describes the verification of an Evidence Record according to the Evidence Record Syntax (ERS, [RFC4998] (Gondrom, T., Brandner, R., and U. Pordesch, “Evidence Record Syntax (ERS),” August 2007.)), using the presented data structure.
An Evidence Record contains a sequence of archiveTimeStampChains which consist of ArchiveTimeStamps. For each archiveTimeStamp the hash algorithm used for the hash tree (digestAlgorithm) and the public key algorithm and hash algorithm in the timestamp signature have to be examined. The relevant date is the time information in the timestamp (date of issue). Starting with the first ArchiveTimestamp it has to be assured that

1. The timestamp uses public key and hash algorithms which have been suitable at the date of issue.

2. The hashtree was build with an hash algorithm that has been suitable at the date of issue as well.

3. Algorithms for timestamp and hashtree in the preceding ArchiveTimestamp must have been suitable at the issuing date of considered ArchiveTimestamp.

4. Algorithms in the last ArchiveTimstamp have to be suitable now.

If the check of one of these items fails, this will lead to a failure of the verification.

A.2. Storing DSSC Policies in Evidence Records (normative) TOC

This section describes how to store a policy in an Evidence Record. ERS provides the field cryptoInfos for the storage of additional verification data. For the integration of a security suitability policy in an Evidence Record the following content types are defined for both ASN.1 and XML representation:

```
DSSC_ASN1 {iso(1) identified-organization(3) dod(6)
        internet(1) security(5) mechanisms(5)
        ltans(11) id-ct(1) id-ct-dssc-asn1(2) }


DSSC_XML {iso(1) identified-organization(3) dod(6)
        internet(1) security(5) mechanisms(5)
        ltans(11) id-ct(1) id-ct-dssc-xml(3) }
```

Appendix B. XML schema (normative)

TOC

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:dssc="http://www.sit.fraunhofer.de/dssc"
           xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
           targetNamespace="http://www.sit.fraunhofer.de/dssc"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
             schemaLocation="http://www.w3.org/2001/xml.xsd"/>
  <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
             schemaLocation="xmldsig-core-schema.xsd"/>
  <xs:element name="SecuritySuitabilityPolicy"
              type="dssc:SecuritySuitabilityPolicyType"/>
  <xs:complexType name="SecuritySuitabilityPolicyType">
    <xs:sequence>
      <xs:element ref="dssc:PolicyName"/>
      <xs:element ref="dssc:Publisher"/>
      <xs:element name="PolicyIssueDate" type="xs:dateTime"/>
      <xs:element name="NextUpdate" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="Usage" type="xs:string" minOccurs="0"/>
      <xs:element ref="dssc:Algorithm" maxOccurs="unbounded"/>
      <xs:element ref="ds:Signature" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="version" type="xs:string" default="1"/>
    <xs:attribute name="id" type="xs:ID"/>
  </xs:complexType>
  <xs:element name="PolicyName" type="dssc:PolicyNameType"/>
  <xs:complexType name="PolicyNameType">
    <xs:sequence>
      <xs:element ref="dssc:Name"/>
      <xs:element ref="dssc:URI" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Publisher" type="dssc:PublisherType"/>
  <xs:complexType name="PublisherType">
    <xs:sequence>
      <xs:element ref="dssc:Name"/>
      <xs:element ref="dssc:Address" minOccurs="0"/>
      <xs:element ref="dssc:URI" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Name" type="xs:string"/>
  <xs:element name="URI" type="xs:anyURI"/>
  <xs:element name="Address" type="dssc:AddressType"/>
  <xs:complexType name="AddressType">
    <xs:sequence>
      <xs:element name="Street" type="xs:string"/>
```

```
    <xs:element name="Locality" type="xs:string"/>
    <xs:element name="StateOrProvince" type="xs:string"
                minOccurs="0"/>
    <xs:element name="PostalCode" type="xs:string"/>
    <xs:element name="Country" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
<xs:element name="Algorithm" type="dssc:AlgorithmType"/>
<xs:complexType name="AlgorithmType">
  <xs:sequence>
    <xs:element ref="dssc:AlgorithmIdentifier"/>
    <xs:element ref="dssc:Evaluation" maxOccurs="unbounded"/>
    <xs:element ref="dssc:Information" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:element name="AlgorithmIdentifier"
            type="dssc:AlgorithmIdentifierType"/>
<xs:complexType name="AlgorithmIdentifierType">
  <xs:sequence>
    <xs:element ref="dssc:Name"/>
    <xs:element name="ObjectIdentifier" type="xs:string"
                maxOccurs="unbounded"/>
    <xs:element ref="dssc:URI" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<xs:element name="Validity" type="dssc:ValidityType"/>
<xs:complexType name="ValidityType">
  <xs:sequence>
    <xs:element name="Start" type="xs:date" minOccurs="0"/>
    <xs:element name="End" type="xs:date" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:element name="Information" type="dssc:InformationType"/>
<xs:complexType name="InformationType">
  <xs:sequence>
    <xs:element name="Text" maxOccurs="unbounded">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string">
            <xs:attribute name="lang"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:element name="Evaluation" type="dssc:EvaluationType"/>
<xs:complexType name="EvaluationType">
  <xs:sequence>
```

```
      <xs:element ref="dssc:Parameter" minOccurs="0"
                                       maxOccurs="unbounded"/>
      <xs:element ref="dssc:Validity"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Parameter" type="dssc:ParameterType"/>
  <xs:complexType name="ParameterType">
    <xs:choice>
      <xs:element name="Exact" type="xs:string"/>
      <xs:element ref="dssc:Min"/>
      <xs:element ref="dssc:Max"/>
      <xs:element name="Range">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="dssc:Min"/>
            <xs:element ref="dssc:Max"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:any namespace="##other"/>
    </xs:choice>
    <xs:attribute name="name" type="xs:string" use="required"/>
  </xs:complexType>
  <xs:element name="Min" type="xs:string"/>
  <xs:element name="Max" type="xs:string"/>
</xs:schema>
```

Appendix C. ASN.1 Module in 1988 Syntax (informative)

TOC

```
ASN.1-Module
```

```
DSSC {iso(1) identified-organization(3) dod(6)
         internet(1) security(5) mechanisms(5)
         ltans(11) id-mod(0) id-mod-dssc88(6) id-mod-dssc88-v1(1) }

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

-- EXPORT ALL --

IMPORTS

-- Import from RFC 5280 [RFC5280]
-- Delete following import statement
-- if "new" types are supported

UTF8String FROM PKIX1Explicit88
            { iso(1) identified-organization(3) dod(6)
            internet(1) security(5) mechanisms(5) pkix(7)
            mod(0) pkix1-explicit(18) }


-- Import from RFC 3852 [RFC3852]

ContentInfo FROM CryptographicMessageSyntax2004
            { iso(1) member-body(2) us(840)
            rsadsi(113549) pkcs(1) pkcs-9(9)
            smime(16) modules(0) cms-2004(24)}

;

SecuritySuitabilityPolicy ::= ContentInfo

-- contentType is id-signedData as defined in [RFC3852]
-- content is SignedData as defined in [RFC3852]
-- eContentType within SignedData is id-ct-dssc
-- eContent within SignedData is TBSPolicy

id-ct-dssc  OBJECT IDENTIFIER ::= {
            iso(1) identified-organization(3) dod(6)
            internet(1) security(5) mechanisms(5)
            ltans(11) id-ct(1) id-ct-dssc-tbsPolicy(6) }

TBSPolicy ::= SEQUENCE {
     version          INTEGER { v1(1) }      OPTIONAL,
     policyName       PolicyName,
     publisher        Publisher,
     policyIssueDate  GeneralizedTime,
     nextUpdate       GeneralizedTime        OPTIONAL,
```

```
     usage            UTF8String             OPTIONAL,
     algorithms       SEQUENCE OF Algorithm
}

PolicyName ::= SEQUENCE {
     name  UTF8String,
     oid   OBJECT IDENTIFIER OPTIONAL
}

Publisher ::= SEQUENCE {
     name        UTF8String,
     address [0] Address     OPTIONAL,
     uri     [1] IA5String   OPTIONAL
}

Address ::= SEQUENCE {
     street           [0] UTF8String,
     locality         [1] UTF8String,
     stateOrProvince  [2] UTF8String OPTIONAL,
     postalCode       [3] UTF8String,
     country          [4] UTF8String
}

Algorithm ::= SEQUENCE {
     algorithmIdentifier     AlgID,
     evaluations             SEQUENCE OF Evaluation,
     information         [0] SEQUENCE OF UTF8String OPTIONAL
}

AlgID ::= SEQUENCE {
     name      UTF8String,
     oid   [0] SEQUENCE OF OBJECT IDENTIFIER,
     uri   [1] SEQUENCE OF IA5String OPTIONAL
}

Evaluation ::= SEQUENCE {
     parameters           [0] SEQUENCE OF Parameter  OPTIONAL,
     validity             [1] Validity
}

Parameter ::= SEQUENCE {
     name        UTF8String,
     constraint  CHOICE {
                   exact  [0] OCTET STRING,
                   min    [1] OCTET STRING,
                   max    [2] OCTET STRING,
                   range  [3] Range,
                   other  [4] OtherConstraints
     }
```

```
}

OtherConstraints ::= SEQUENCE {
     otherConstraintType  OBJECT IDENTIFIER,
     otherConstraint      ANY DEFINED BY otherConstraintType
}

Range ::= SEQUENCE {
     min  [0] OCTET STRING,
     max  [1] OCTET STRING
}

Validity ::= SEQUENCE {
     start  [0] GeneralizedTime OPTIONAL,
     end    [1] GeneralizedTime OPTIONAL
}

END
```

Appendix D. ASN.1 Module in 1997 Syntax (normative)

TOC

ASN.1-Module

```
DSSC {iso(1) identified-organization(3) dod(6)
         internet(1) security(5) mechanisms(5)
         ltans(11) id-mod(0) id-mod-dssc(7) id-mod-dssc-v1(1) }

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

-- EXPORT ALL --

IMPORTS

-- Import from RFC 5280 [RFC5280]
-- Delete following import statement
-- if "new" types are supported

UTF8String FROM PKIX1Explicit88
            { iso(1) identified-organization(3) dod(6)
            internet(1) security(5) mechanisms(5) pkix(7)
            mod(0) pkix1-explicit(18) }


-- Import from RFC 3852 [RFC3852]

ContentInfo FROM CryptographicMessageSyntax2004
            { iso(1) member-body(2) us(840)
            rsadsi(113549) pkcs(1) pkcs-9(9)
            smime(16) modules(0) cms-2004(24)}

;

SecuritySuitabilityPolicy ::= ContentInfo

-- contentType is id-signedData as defined in [RFC3852]
-- content is SignedData as defined in [RFC3852]
-- eContentType within SignedData is id-ct-dssc
-- eContent within SignedData is TBSPolicy

id-ct-dssc  OBJECT IDENTIFIER ::= {
            iso(1) identified-organization(3) dod(6)
            internet(1) security(5) mechanisms(5)
            ltans(11) id-ct(1) id-ct-dssc-tbsPolicy(6) }

TBSPolicy ::= SEQUENCE {
     version          INTEGER { v1(1) }      OPTIONAL,
     policyName       PolicyName,
     publisher        Publisher,
     policyIssueDate  GeneralizedTime,
     nextUpdate       GeneralizedTime        OPTIONAL,
```

```
     usage            UTF8String             OPTIONAL,
     algorithms       SEQUENCE OF Algorithm
}

PolicyName ::= SEQUENCE {
     name  UTF8String,
     oid   OBJECT IDENTIFIER OPTIONAL
}

Publisher ::= SEQUENCE {
     name         UTF8String,
     address  [0] Address     OPTIONAL,
     uri      [1] IA5String   OPTIONAL
}

Address ::= SEQUENCE {
     street           [0] UTF8String,
     locality         [1] UTF8String,
     stateOrProvince  [2] UTF8String OPTIONAL,
     postalCode       [3] UTF8String,
     country          [4] UTF8String
}

Algorithm ::= SEQUENCE {
     algorithmIdentifier     AlgID,
     evaluations             SEQUENCE OF Evaluation,
     information         [0] SEQUENCE OF UTF8String OPTIONAL
}

AlgID ::= SEQUENCE {
     name      UTF8String,
     oid   [0] SEQUENCE OF OBJECT IDENTIFIER,
     uri   [1] SEQUENCE OF IA5String         OPTIONAL
}

Evaluation ::= SEQUENCE {
     parameters           [0] SEQUENCE OF Parameter  OPTIONAL,
     validity             [1] Validity
}

Parameter ::= SEQUENCE {
     name        UTF8String,
     constraint  CHOICE {
                    exact  [0] OCTET STRING,
                    min    [1] OCTET STRING,
                    max    [2] OCTET STRING,
                    range  [3] Range,
                    other  [4] OtherConstraints
     }
```

```
}

OtherConstraints ::= SEQUENCE {
     otherConstraintType  CONSTRAINT-TYPE.&id ({SupportedConstraints}),
     otherConstraint      CONSTRAINT-TYPE.&Type
                         ({SupportedConstraints}{@otherConstraintType})
}

CONSTRAINT-TYPE ::= TYPE-IDENTIFIER

SupportedConstraints CONSTRAINT-TYPE ::= {...}

Range ::= SEQUENCE {
     min  [0] OCTET STRING,
     max  [1] OCTET STRING
}

Validity ::= SEQUENCE {
     start  [0] GeneralizedTime OPTIONAL,
     end    [1] GeneralizedTime OPTIONAL
}

END
```

Appendix E. Example

TOC

In the following an example of a policy is presented. It is generated on the basis of an evaluation of the German Federal Network Agency ([BNetzAg.2008] (Federal Network Agency for Electricity, Gas, Telecommunications, Post and Railway, “Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Übersicht über geeignete Algorithmen),” December 2007.)). The policy consists on hash algorithms as well as public key algorithms. RSA with modulus length of 768 is an example for an expired algorithm.

```
<SecuritySuitabilityPolicy xmlns="http://www.sit.fraunhofer.de/dssc"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <PolicyName>
    <Name>Evaluation of suitable signature algorithms 2008</Name>
  </PolicyName>
  <Publisher>
    <Name>Federal Network Agency</Name>
  </Publisher>
  <PolicyIssueDate>2007-12-17T00:00:00</PolicyIssueDate>
  <Usage>Qualified electronic signatures</Usage>
  <Algorithm>
    <AlgorithmIdentifier>
      <Name>SHA-1</Name>
      <ObjectIdentifier>1.3.14.3.2.26</ObjectIdentifier>
    </AlgorithmIdentifier>
    <Evaluation>
      <Validity>
        <End>2008-06-31</End>
      </Validity>
    </Evaluation>
  </Algorithm>
  <Algorithm>
    <AlgorithmIdentifier>
      <Name>RIPEMD-160</Name>
      <ObjectIdentifier>1.3.36.3.2.1</ObjectIdentifier>
    </AlgorithmIdentifier>
    <Evaluation>
      <Validity>
        <End>2010-12-31</End>
      </Validity>
    </Evaluation>
  </Algorithm>
  <Algorithm>
    <AlgorithmIdentifier>
      <Name>SHA-224</Name>
      <ObjectIdentifier>2.16.840.1.101.3.4.2.4</ObjectIdentifier>
    </AlgorithmIdentifier>
    <Evaluation>
      <Validity>
        <End>2014-12-31</End>
      </Validity>
    </Evaluation>
  </Algorithm>
  <Algorithm>
    <AlgorithmIdentifier>
      <Name>SHA-256</Name>
      <ObjectIdentifier>2.16.840.1.101.3.4.2.1</ObjectIdentifier>
```

```
  </AlgorithmIdentifier>
  <Evaluation>
    <Validity>
      <End>2014-12-31</End>
    </Validity>
  </Evaluation>
</Algorithm>
<Algorithm>
  <AlgorithmIdentifier>
    <Name>SHA-384</Name>
    <ObjectIdentifier>2.16.840.1.101.3.4.2.2</ObjectIdentifier>
  </AlgorithmIdentifier>
  <Evaluation>
    <Validity>
      <End>2014-12-31</End>
    </Validity>
  </Evaluation>
</Algorithm>
<Algorithm>
  <AlgorithmIdentifier>
    <Name>SHA-512</Name>
    <ObjectIdentifier>2.16.840.1.101.3.4.2.3</ObjectIdentifier>
  </AlgorithmIdentifier>
  <Evaluation>
    <Validity>
      <End>2014-12-31</End>
    </Validity>
  </Evaluation>
</Algorithm>
<Algorithm>
  <AlgorithmIdentifier>
    <Name>RSA</Name>
    <ObjectIdentifier>1.2.840.113549.1.1.1</ObjectIdentifier>
  </AlgorithmIdentifier>
  <Evaluation>
    <Parameter name="moduluslength">
      <Min>768</Min>
    </Parameter>
    <Validity>
      <End>2000-12-31</End>
    </Validity>
  </Evaluation>
  <Evaluation>
    <Parameter name="moduluslength">
      <Min>1024</Min>
    </Parameter>
    <Validity>
      <End>2008-03-31</End>
    </Validity>
```

```
  </Evaluation>
  <Evaluation>
    <Parameter name="moduluslength">
      <Min>1280</Min>
    </Parameter>
    <Validity>
      <End>2008-12-31</End>
    </Validity>
  </Evaluation>
  <Evaluation>
    <Parameter name="moduluslength">
      <Min>1536</Min>
    </Parameter>
    <Validity>
      <End>2009-12-31</End>
    </Validity>
  </Evaluation>
  <Evaluation>
    <Parameter name="moduluslength">
      <Min>1728</Min>
    </Parameter>
    <Validity>
      <End>2010-12-31</End>
    </Validity>
  </Evaluation>
  <Evaluation>
    <Parameter name="moduluslength">
      <Min>1976</Min>
    </Parameter>
    <Validity>
      <End>2014-12-31</End>
    </Validity>´
  </Evaluation>
  <Evaluation>
    <Parameter name="moduluslength">
      <Min>2048</Min>
    </Parameter>
    <Validity>
      <End>2014-12-31</End>
    </Validity>
  </Evaluation>
</Algorithm>
<Algorithm>
  <AlgorithmIdentifier>
    <Name>DSA</Name>
    <ObjectIdentifier>1.2.840.10040.4.1</ObjectIdentifier>
  </AlgorithmIdentifier>
  <Evaluation>
    <Parameter name="plength">
```

```
      <Min>1024</Min>
    </Parameter>
    <Parameter name="qlength">
      <Min>160</Min>
    </Parameter>
    <Validity>
      <End>2007-12-31</End>
    </Validity>
  </Evaluation>
  <Evaluation>
    <Parameter name="plength">
      <Min>1280</Min>
    </Parameter>
    <Parameter name="qlength">
      <Min>160</Min>
    </Parameter>
    <Validity>
      <End>2008-12-31</End>
    </Validity>
  </Evaluation>
  <Evaluation>
    <Parameter name="plength">
      <Min>1536</Min>
    </Parameter>
    <Parameter name="qlength">
      <Min>160</Min>
    </Parameter>
    <Validity>
      <End>2009-12-31</End>
    </Validity>
  </Evaluation>
  <Evaluation>
    <Parameter name="plength">
      <Min>2048</Min>
    </Parameter>
    <Parameter name="qlength">
      <Min>160</Min>
    </Parameter>
    <Validity>
      <End>2009-12-31</End>
    </Validity>
  </Evaluation>
  <Evaluation>
    <Parameter name="plength">
      <Min>2048</Min>
    </Parameter>
    <Parameter name="qlength">
      <Min>224</Min>
    </Parameter>
```

```
        <Validity>
          <End>2014-12-31</End>
        </Validity>
      </Evaluation>
    </Algorithm>
  </SecuritySuitabilityPolicy>
```

Combined algorithms should also be part of the policy since some programs know the object identifiers of combined algorithms instead of the general public key algorithm. The following excerpt describes a combined algorithm. The validity end date is given by the end dates of RSA and RIPEMD-160, in particular it is the former one. Combined algorithms could replace the public key algorithms in the policy example. They could also be listed together with public key algorithms.

```
  <Algorithm>
    <AlgorithmIdentifier>
      <Name>RIPEMD-160 with RSA 2048</Name>
      <ObjectIdentifier>1.3.36.3.3.1.2</ObjectIdentifier>
    </AlgorithmIdentifier>
    <Evaluation>
      <Parameter name="moduluslength">
        <Min>2048</Min>
      </Parameter>
      <Validity>
        <End>2010-12-31</End>
      </Validity>
    </Evaluation>
  </Algorithm>
```

Appendix F. Disclaimer

TOC

Authors' Addresses

TOC

	Thomas Kunz
	Fraunhofer Institute for Secure Information Technology
	Rheinstrasse 75
	Darmstadt D-64295
	Germany
Email:	thomas.kunz@sit.fraunhofer.de
	Susanne Okunick
	pawisda systems GmbH
	Robert-Koch-Strasse 9
	Weiterstadt D-64331
	Germany
Email:	susanne.okunick@pawisda.de
	Ulrich Pordesch
	Fraunhofer Gesellschaft
	Rheinstrasse 75
	Darmstadt D-64295
	Germany
Email:	ulrich.pordesch@zv.fraunhofer.de