

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: May 19, 2012

J. Schaad  
Soaring Hawk Consulting  
November 16, 2011

**S/MIME Capabilities for Public Key Definitions**  
**draft-ietf-pkix-pubkey-caps-04**

Abstract

This document defines a set of S/MIME Capability types for ASN.1 encoding for the current set of public keys define in the PKIX working group. This facilitates the ability for a requester to specify information on the public keys and signature algorithms to be used in responses. An example of where this is used is with the OSCP Agility draft.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

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

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

This Internet-Draft will expire on May 19, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as

described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">3</a>
<a href="#">1.1.</a>	ASN.1 Notation . . . . .	<a href="#">3</a>
<a href="#">1.2.</a>	Requirements Terminology . . . . .	<a href="#">4</a>
<a href="#">2.</a>	RSA Public Keys . . . . .	<a href="#">5</a>
<a href="#">2.1.</a>	Generic RSA Public Keys . . . . .	<a href="#">5</a>
<a href="#">2.2.</a>	RSASSA-PSS Signature Public Keys . . . . .	<a href="#">6</a>
<a href="#">2.3.</a>	RSA ES-OAEP Key Transport Public Keys . . . . .	<a href="#">6</a>
<a href="#">3.</a>	Diffie-Hellman Keys . . . . .	<a href="#">8</a>
<a href="#">3.1.</a>	DSA Signature Public Key . . . . .	<a href="#">8</a>
<a href="#">3.2.</a>	DH Key Agreement Keys . . . . .	<a href="#">9</a>
<a href="#">4.</a>	Elliptical Curve Keys . . . . .	<a href="#">10</a>
<a href="#">4.1.</a>	Generic Elliptical Curve Keys . . . . .	<a href="#">10</a>
<a href="#">4.2.</a>	Elliptical Curve DH Keys . . . . .	<a href="#">11</a>
<a href="#">4.3.</a>	Elliptical Curve MQV Keys . . . . .	<a href="#">11</a>
<a href="#">5.</a>	RSASSA-PSS Signature Algorithm Capability . . . . .	<a href="#">12</a>
<a href="#">6.</a>	Security Considerations . . . . .	<a href="#">14</a>
<a href="#">7.</a>	IANA Considerations . . . . .	<a href="#">15</a>
<a href="#">8.</a>	References . . . . .	<a href="#">16</a>
<a href="#">8.1.</a>	Normative References . . . . .	<a href="#">16</a>
<a href="#">8.2.</a>	Informative References . . . . .	<a href="#">16</a>
<a href="#">Appendix A.</a>	2008 ASN.1 Module . . . . .	<a href="#">17</a>
<a href="#">Appendix B.</a>	1988 ASN.1 Module . . . . .	<a href="#">20</a>
<a href="#">Appendix C.</a>	Future Work . . . . .	<a href="#">22</a>
	Author's Address . . . . .	<a href="#">23</a>



## **1. Introduction**

In the process of dealing with the OSCP agility issues in [\[RFC6277\]](#) it was noted that we really wanted to describe information to be used in selecting a public key, but we did not currently have any way of doing so at the current time. This document fills that hole by defining a set of S/MIME Capability types for a small set of public key representations.

S/MIME Capabilities were originally defined in [\[SMIMEv3-MSG\]](#) as a way for the sender of an S/MIME message to tell the recipient of the message the set of encryption algorithms that were supported by the senders system. In the beginning, the focus was primarily on communicating the set of encryption algorithms that were supported. Over time it was expanded to allow for an S/MIME client to say that it supported the compression data type and binary contents. As originally defined it was targeted toward supporting items with a small number of possible parameters. For the RC2 encryption algorithm only two values from the entire range of values were ever used. The object of restricting the set of values was so that a client could do a simple binary comparison without having to decode the S/MIME capability. This was especially easy since most just consisted of the object identifier for the algorithm.

Given that we are assigning different data types to the algorithm descriptors here, and many of the algorithm descriptors are the same as are used in signature, key transport or key agreement algorithms, the public key versions of these structures MUST NOT be placed in the same locations as the other versions. It is expected that the places where one needs S/MIME capabilities for public keys is going to be vastly different than for the other values.

### **1.1. ASN.1 Notation**

The main body of the text is written using snippets of ASN.1 that are extracted from the ASN.1 2008 module in [Appendix A](#). This is because I am a strong advocate of moving to the current versions of ASN.1 as they can contain meta-data which is not representable in the 1988 version of ASN.1. In keeping with the current policy of the PKIX working group, the 1988 module is still to be considered the normative module in the event of a conflict between the contents of the two modules.

When reading this document, it is assumed that you will have a degree of familiarity with the basic object module that is presented in [section 3 of RFC 5912](#) ([\[RFC5912\]](#)). We use the SMIME-CAPS object in this document, it associates two fields together in a single object.



```
SMIME-CAPS ::= CLASS {  
    &id          OBJECT IDENTIFIER UNIQUE,  
    &Type        OPTIONAL  
}  
WITH SYNTAX { [TYPE &Type] IDENTIFIED BY &id }
```

These fields are:

&id contains an object identifier. When placed in an object set, this element is tagged so that no two elements can be placed in the set that have the same value in the &id field. Note that this is not a restriction which says that only a single object can exist with a single object identifier.

&Type optionally contains an ASN.1 type identifier. If the field &Type is not defined then the optional parameters field of the AlgorithmIdentifier type would be omitted.

The class also has a specialized syntax for how to define an object in this class. The all upper case words TYPE IDENTIFIER and BY are syntactic sugar to make it easier to read. The square brackets defined optional pieces of the syntax.

One of the things that can be done is to reference the fields of an object while defining other objects. This means that if an object called foo has a field named &value, the value can be directly referenced as foo.&value. This means that we automatically get any updates to values or types and we do not need to do any replication of the data.

## **1.2. Requirements Terminology**

When capitalized 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 [\[RFC2119\]](#).



## **2. RSA Public Keys**

There are currently three different public key object identifiers for RSA public keys. These are RSA, RSA ES-OAEP and RSA SSA-PSS.

### **2.1. Generic RSA Public Keys**

Almost all RSA keys that are contained in certificates today use the generic RSA public key format and identifier. This allows for the public key to be used both for key transport and for signature validation (assuming it is compatible with the bits in the key usage extension). The only reason for using one of more specific public key identifiers is if the user wants to restrict the usage of the RSA public key with a specific algorithm.

For the generic RSA public key, the S/MIME capability that is advertised is a request for a specific key size to be used. This would normally be used for dealing with a request on the key to be used for a signature that the client would then verify. In general the user would provide a specific key when a key transport algorithm is being considered.

The ASN.1 that is used for the generic RSA public key is defined as below:

```
scap-pk-rsa SMIME-CAPS ::= {  
    TYPE RSAKeyCapabilities  
    IDENTIFIED BY pk-rsa.&id  
}  
  
RSAKeyCapabilities ::= SEQUENCE {  
    minKeySize      RSAKeySize,  
    maxKeySize      RSAKeySize OPTIONAL  
}  
  
RSAKeySize ::= INTEGER (1024 | 2048 | 3072 | 7680 | 15360 |  
                        4096 | 8192, ...)
```

In the above ASN.1 we have defined the following:

scap-pk-rsa is a new SMIME-CAP object. This object associates the existing object identifier (rsaEncryption) used for the public key in certificates (defined in [[RFC3279](#)] and [[RFC5912](#)]) with a new type defined in this document.





`RSACapabilities` carries the set of desired capabilities for an RSA key. The fields of this type are:

`minKeySize` contains the minimum length of the RSA modulus to be used. This field SHOULD NOT contain a value less than 1024.

`maxKeySize` contains the maximum length of the RSA modules that should be used. If this field is absent then no maximum length is requested/expected. This value is normally selected so as not to cause the current code to run unacceptably long when processing signatures.

`RSACapabilities` provides a set of suggested values to be used. The values 1024, 2048, 3072, 7680 and 15360 are from the NIST guide on signature sizes [[NIST-SIZES](#)] while the others are common powers of two that would be used. The list is not closed and other values can be used.

## **[2.2.](#) RSASSA-PSS Signature Public Keys**

While most of the time one will use the generic RSA public key identifier in a certificate, the RSA SS-PPS identifier can be used if the owner of the key desires to restrict the usage of the key to just this algorithm. This algorithm does have the ability to place a set of algorithm parameters in the public key info structure, they have not been included in this location as the same information should be carried in the signature S/MIME capabilities instead.

The ASN.1 that is used for the RSA SS-PPS public key is defined below:

```
scap-pk-rsaSSA-PSS SMIME-CAPS ::= {  
  TYPE RSACapabilities  
  IDENTIFIED BY pk-rsaSSA-PSS.&id  
}
```

In the above ASN.1 we have defined the following:

`scap-pk-rsaSSA-PSS` is a new SMIME-CAP object. This object associates the existing object identifier (`id-RSASSA-PSS`) used for the public key certificates (defined in [[RFC4055](#)] and [[RFC5912](#)]) with type `RSACapabilities`.

## **[2.3.](#) RSA ES-OAEP Key Transport Public Keys**

While most of the time one will use the generic RSA public key identifier in a certificate, the RSA ES-OAEP identifier can be used if the owner of the key desires to restrict the usage of the key to



just this algorithm. This algorithm does have the ability to place a set of algorithm parameters in the public key info structure, they have not been included in this location as the same information should be carried in the key transport S/MIME capabilities instead.

The ASN.1 that is used for the RSA ES-OAEP public key is defined below:

```
scap-pk-rsaES-OAEP SMIME-CAPS ::= {  
    TYPE RSAKeyCapabilities  
    IDENTIFIED BY pk-rsaES-OAEP.&id  
}
```

In the above ASN.1 we have defined the following:

scap-pk-rsaES-OAEP is a new SMIME-CAP object. This object associates the existing object identifier (id-RSAES-OAEP) used for the public key certificates (defined in [[RFC4055](#)] and [[RFC5912](#)]) with type RSAKeyCapabilities.



### **3. Diffie-Hellman Keys**

There is current two Diffie-Hellman public key object identifiers. These are DH key agreement and DSA.

#### **3.1. DSA Signature Public Key**

This public key type is used for the validation of DSA signatures.

The ASN.1 that is used for DSA keys is defined below:

```
scap-pk-dsa SMIME-CAPS ::= {  
  TYPE DSAKeyCapabilities  
  IDENTIFIED BY pk-dsa.&id  
}  
  
DSAKeyCapabilities ::= CHOICE {  
  keySizes      [0] SEQUENCE {  
    minKeySize      DSAKeySize,  
    maxKeySize      DSAKeySize OPTIONAL  
  },  
  keyParams      [1] pk-dsa.&Params  
}  
  
DSAKeySize ::= INTEGER (1024 | 2048 | 3072 | 7680 | 15360 )
```

In the above ASN.1 we have defined the following:

scap-pk-dsa is a new SMIME-CAP object. This object associated the existing object identifier (id-dsa) used for the public key in certificates (defined in [[RFC3279](#)] and [[RFC5912](#)]) with a new type defined here, DSAKeyCapabilities.

DSAKeyCapabilities carries the desired set of capabilities for the DSA key. The fields of this type are:

keySizes is used when only a key size is needed to be specified and not a specific group. It is expected that this would be the most commonly used of the two options. In key sizes the fields are used as follows:

minKeySize contains the minimum length of the DSA modulus to be used.

maxKeySize contains the maximum length of the DSA modules that should be used. If this field is absent then no maximum length is requested/expected.



keyParams contains the exact set of DSA for the key used to sign the message. This field is provided for completeness and to match the fields for Elliptical Curve, however it is expected that usage of this field is extremely rare.

### **3.2. DH Key Agreement Keys**

This public key type is used with the Diffie-Hellman key agreement algorithm.

The ASN.1 that is used for DH keys is defined below:

```
scap-pk-dh SMIME-CAPS ::= {  
  TYPE DSAKeyCapabilities  
  IDENTIFIED BY pk-dh.&id  
}
```

In the above ASN.1 we have defined the following:

scap-pk-dh is a new SMIME-CAP object. This object associates the existing object identifier (id-dh) used for the public key algorithm in the certificates (defined in [\[RFC3279\]](#) and [\[RFC5912\]](#)) with a new type defined above, DSAKeyCapabilities.





## **4. Elliptical Curve Keys**

There are currently three Elliptical Curve public key object identifiers. These are EC, EC-DH and EC-MQV

### **4.1. Generic Elliptical Curve Keys**

All most all ECC keys that are contained in certificates today use the generic ECC public key format and identifier. This allows for the public key to be used both for key agreement and for signature validation (assuming the appropriate bits are in the certificate). The only reason for using one of the more specific public key identifier is if the user wants to restrict the usage of the ECC public key with a specific algorithm.

For the generic ECC public key, the S/MIME capability that is advertised is a request for a specific group to be used.

The ASN.1 that is used for the generic ECC public key is defined as below:

```
scap-pk-ec SMIME-CAPS ::= {  
    TYPE EC-SMimeCaps  
    IDENTIFIED BY pk-ec.&id  
}  
  
EC-SMimeCaps ::= SEQUENCE (SIZE (1..MAX)) OF ECPParameters
```

In the above ASN.1 we have defined the following:

scap-pk-ec is a new SMIME-CAP object. This object associated the existing object identifier (id-ecPublicKey) used for the public key algorithm in the certificates (defined in [[RFC3279](#)] and [[RFC5912](#)]) with the new type EC-SMimeCaps.

EC-SMimeCaps carries a sequence of at least one ECPParameters structure. This allows for multiple curves to be requested in a single capability request. A maximum/minimum style of specifying sizes is not provided as much greater care is required in selecting a specific curve than is needed to create the parameters for a DSA/DH key. As specified in [[RFC3279](#)], for PKIX compliant certificates only the namedCurve choice of ECPParameters is expected to be used.



#### **4.2. Elliptical Curve DH Keys**

This public key type is used with the Elliptical Curve Diffie-Hellman key agreement algorithm.

The ASN.1 that is used for EC-DH keys is defined below:

```
scap-pk-ecDH SMIME-CAPS ::= {  
  TYPE EC-SMimeCaps  
  IDENTIFIED BY pk-ecDH.&id  
}
```

In the above ASN.1 we have defined the following:

scap-ec-dh is a new SMIME-CAP object. This object associated the existing object identifier (id-??) used for the public key algorithm in the certificate (defined in [[RFC3279](#)] and [[RFC5912](#)]) with the same type structure used for public keys.

#### **4.3. Elliptical Curve MQV Keys**

This public key type is used with the Elliptical Curve MQV key agreement algorithm.

The ASN.1 that is used for EC-MQV keys is defined below:

```
scap-pk-ecMQV SMIME-CAPS ::= {  
  TYPE EC-SMimeCaps  
  IDENTIFIED BY pk-ecMQV.&id  
}
```

In the above ASN.1 we have defined the following:

scap-ec-MQV is a new SMIME-CAP object. This object associated the existing object identifier (id-??) used for the public key algorithm in the certificate (defined in [[RFC3279](#)] and [[RFC5912](#)]) with the same type structure used for public keys.



## 5. RSASSA-PSS Signature Algorithm Capability

This document defines a new S/MIME Capability for the RSA-SSA-PSS signature algorithm. There already exists one in [\[RFC4055\]](#) where the parameters field is not used.

When the S/MIME group defined a S/MIME Capability for the RSA-SSA-PSS signature algorithm, it was done so in the context of how S/MIME defines and uses S/MIME Capabilities. When placed in an S/MIME message [\[SMIME-MSG\]](#) or in a certificate [\[RFC4262\]](#) it is always placed in a sequence of capabilities. This meant that one can place the identifier for RSA-SSA-PSS in the sequence along with the identifier for MD5, SHA-1 and SHA-256. The assumption was then made that one could compute the matrix of all answers and the publisher would support all elements in the matrix. This has the possibility that the publisher could accidentally publish a point in the matrix that is not supported.

In this situation, there is only a single item that is published. This means that we need to publish all of the associated information along with the identifier for the signature algorithm in a single entity. For this reason we now define a new parameter type to be used as the S/MIME capability type which contains a hash identifier and a mask identifier. The ASN.1 used for this is as follows:

```
scap-sa-rsaSSA-PSS SMIME-CAPS ::= {  
    TYPE RsaSsa-Pss-sig-caps  
    IDENTIFIED BY sa-rsaSSA-PSS.&id  
}  
  
RsaSsa-Pss-sig-caps ::= SEQUENCE {  
    hashAlg SMIMECapability{{ HashAlgorithms }},  
    maskAlg SMIMECapability{{ MaskAlgorithmSet }} OPTIONAL,  
    trailerField INTEGER DEFAULT 1  
}  
  
scap-mf-mgf1 SMIME-CAPS ::= {  
    TYPE SMIMECapability{{ HashAlgorithms }}  
    IDENTIFIED BY id-mgf1  
}  
  
MaskAlgorithmSet SMIME-CAPS ::= {scap-mf-mgf1, ...}
```

In the above ASN.1 we have defined the following:



scap-sa-rsaSSA-PSS is a new SMIME-CAP object. This object associates the existing object identifier (id-RSASSA-PSS) used for the signature algorithm (defined in [[RFC4055](#)] and [[RFC5912](#)]) with the new type RsaSsa-Pss-sig-caps.

RsaSsa-Pss-sig-caps carries the desired set of capabilities for the RSA SSA-PSS signature algorithm. The fields of this type are:

hashAlg contains the S/MIME capability for the hash algorithm we are declaring we support with the RSA-SSA-PSS signature algorithm.

maskAlg contains the S/MIME capability for the mask algorithm we are declaring we support with the RSA-SSA-PSS signature algorithm.

trailerField specifies which trailer field algorithm is being supported. This MUST be the value 1.

NOTE: In at least one iteration of the design we used a sequence of hash identifiers and a sequence of masking functions and again made the assumption that entire matrix would be supported. This has been removed at this point since the original intent of S/MIME capabilities is that one should be able to do a binary comparison of the DER encoding of the field and determine a specific capability was published. We could return back to using the sequence if we wanted to lose the ability to do a binary compare but needed to shorten the encodings. This does not currently appear to be an issue at this point.





## 6. Security Considerations

This document provides new fields that can be placed in an S/MIME capabilities sequence. There are number of considerations that need to be taking into account when doing this.

As mentioned above, there are a small number of cases where the same object identifier may be used to identify a public key and an algorithm. This is the case for many years with the OID `rsaEncryption` where it identifies both a public key and the RSA v1.5 key transport algorithm. This means that when an S/MIME capabilities sequence is defined care needs to be taken to specify the types of algorithms and/or public keys that are to be specified in that sequence. In general, it is expected that algorithms and public keys will be segregated.

The more detailed the information that is communicated, the better the end results are going to be. If you can state you do RSA v1.5, EC-DSA, SHA-1 and SHA-256, then it would imply that all four values are supported. It may be however that EC-DSA with SHA-1 is not supported. Not including the SHA-1 hash algorithm could lead to problems as RSA with SHA-1 could be the only point of intersection, but including it means that a result may be returned that cannot be processed.

The more information passed the better. The more choices that are passed, the better the odds that both parties will be able to agree on a common algorithm.

The less information passed the better. Passing too much information can lead to computational issues in trying to deal with the possibilities. This becomes acute when a negotiation over algorithms is going on between multiple parties (such as sending an encrypted S/MIME message) where the amount of memory and processing time can be greatly expanded if there are a large number of choices for each recipient.

Ordering of preference of algorithms is not always supported by all places where S/MIME capabilities are used. The addition of preference ordering greatly complicates the decisions to be used, especially as it is expected that not all parties will agree on the same ordering.



## **7. IANA Considerations**

This document has no IANA considerations.

## **8. References**

### **8.1. Normative References**

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC3279] Bassham, L., Polk, W., and R. Housley, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", [RFC 3279](#), April 2002.
- [RFC4055] Schaad, J., Kaliski, B., and R. Housley, "Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", [RFC 4055](#), June 2005.

### **8.2. Informative References**

- [RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", [RFC 5912](#), June 2010.
- [RFC6277] Santesson, S. and P. Hallam-Baker, "Online Certificate Status Protocol Algorithm Agility", [RFC 6277](#), June 2011.
- [SMIME-MSG]  
Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", [RFC 5751](#), January 2010.
- [RFC4262] Santesson, S., "X.509 Certificate Extension for Secure/Multipurpose Internet Mail Extensions (S/MIME) Capabilities", [RFC 4262](#), December 2005.
- [SMIMEv3-MSG]  
Ramsdell, B., "S/MIME Version 3 Message Specification", [RFC 2633](#), June 1999.
- [NIST-SIZES]  
Barker, E., Barker, W., Burr, W., Polk, W., and M. Smid, "Recommendation for Key Management -- Part 1: General", NIST Special Publication 800-57, March 2007.



**Appendix A. 2008 ASN.1 Module**

This appendix contains a module compatible with the work done to update the PKIX ASN.1 modules to recent versions of the ASN.1 specifications [RFC5912]. This appendix is to be considered informational per the current direction of the PKIX working group.

**PUBLIC-KEY-SMIME-CAPABILITIES**

```
{ iso(1) identified-organization(3) dod(6) internet(1)
  security(5) mechanisms(5) pkix(7) id-mod(0) TBD5 }
```

DEFINITIONS ::=

BEGIN

IMPORTS

SMIME-CAPS, PUBLIC-KEY, SMIMECapability

FROM AlgorithmInformation-2009

```
{ iso(1) identified-organization(3) dod(6) internet(1)
  security(5) mechanisms(5) pkix(7) id-mod(0)
  id-mod-algorithmInformation-02(58)}
```

pk-rsa, pk-dsa, pk-dh, pk-ec, pk-ecDH, pk-ecMQV, ECPParameters

FROM PKIXAlgs-2009

```
{ iso(1) identified-organization(3) dod(6) internet(1)
  security(5) mechanisms(5) pkix(7) id-mod(0)
  id-mod-pkix1-algorithms2008-02(56) }
```

pk-rsaSSA-PSS, pk-rsaES-OAEP, sa-rsaSSA-PSS,

HashAlgorithms, id-mgf1

FROM PKIX1-PSS-OAEP-Algorithms-2009

```
{ iso(1) identified-organization(3) dod(6) internet(1)
  security(5) mechanisms(5) pkix(7) id-mod(0)
  id-mod-pkix1-rsa-pkalgs-02(54)}
```

;

--

-- Define a set containing all of the S/MIME capabilities defined  
-- by this document

--

SMimeCaps SMIME-CAPS ::= {

```
  PubKeys-SMimeCaps |
  scap-sa-rsaSSA-PSS
```

}

PubKeys-SMimeCaps SMIME-CAPS ::= {

```
  scap-pk-rsa | scap-pk-rsaSSA-PSS |
  scap-pk-dsa |
  scap-pk-ec | scap-pk-ecDH
```

}





```
--
-- We defined RSA keys from the modules RFC3279 and RFC4055
--

scap-pk-rsa SMIME-CAPS ::= {
  TYPE RSAKeyCapabilities
  IDENTIFIED BY pk-rsa.&id
}

RSAKeyCapabilities ::= SEQUENCE {
  minKeySize      RSAKeySize,
  maxKeySize      RSAKeySize OPTIONAL
}

RSAKeySize ::= INTEGER (1024 | 2048 | 3072 | 7680 | 15360 |
                        4096 | 8192, ...)

scap-pk-rsaES-OAEP SMIME-CAPS ::= {
  TYPE RSAKeyCapabilities
  IDENTIFIED BY pk-rsaES-OAEP.&id
}

scap-pk-rsaSSA-PSS SMIME-CAPS ::= {
  TYPE RSAKeyCapabilities
  IDENTIFIED BY pk-rsaSSA-PSS.&id
}

scap-sa-rsaSSA-PSS SMIME-CAPS ::= {
  TYPE RsaSsa-Pss-sig-caps
  IDENTIFIED BY sa-rsaSSA-PSS.&id
}

RsaSsa-Pss-sig-caps ::= SEQUENCE {
  hashAlg  SMIMECapability{{ HashAlgorithms }},
  maskAlg  SMIMECapability{{ MaskAlgorithmSet }} OPTIONAL,
  trailerField INTEGER DEFAULT 1
}

scap-mf-mgf1 SMIME-CAPS ::= {
  TYPE SMIMECapability{{ HashAlgorithms }}
  IDENTIFIED BY id-mgf1
}

MaskAlgorithmSet SMIME-CAPS ::= {scap-mf-mgf1, ...}

--
-- we define DH/DSA keys from the module RFC3279
```



```
--

scap-pk-dsa SMIME-CAPS ::= {
  TYPE DSAKeyCapabilities
  IDENTIFIED BY pk-dsa.&id
}

DSAKeyCapabilities ::= CHOICE {
  keySizes      [0] SEQUENCE {
    minKeySize      DSAKeySize,
    maxKeySize      DSAKeySize OPTIONAL
  },
  keyParams      [1] pk-dsa.&Params
}

DSAKeySize ::= INTEGER (1024 | 2048 | 3072 | 7680 | 15360 )

scap-pk-dh SMIME-CAPS ::= {
  TYPE DSAKeyCapabilities
  IDENTIFIED BY pk-dh.&id
}

--
-- we define Elliptical Curve keys from the module RFC3279
--

scap-pk-ec SMIME-CAPS ::= {
  TYPE EC-SMimeCaps
  IDENTIFIED BY pk-ec.&id
}

EC-SMimeCaps ::= SEQUENCE (SIZE (1..MAX)) OF ECParameters

scap-pk-ecDH SMIME-CAPS ::= {
  TYPE EC-SMimeCaps
  IDENTIFIED BY pk-ecDH.&id
}

scap-pk-ecMQV SMIME-CAPS ::= {
  TYPE EC-SMimeCaps
  IDENTIFIED BY pk-ecMQV.&id
}

END
```



**Appendix B. 1988 ASN.1 Module**

This appendix contains the normative ASN.1 module for this document.

PUBLIC-KEY-SMIME-CAPABILITIES-88

```
{ iso(1) identified-organization(3) dod(6) internet(1)
  security(5) mechanisms(5) pkix(7) id-mod(0) TBD5 }
```

DEFINITIONS ::=

BEGIN

IMPORTS

ECParameters

FROM PKIX1Algorithms88

```
{ iso(1) identified-organization(3) dod(6)
  internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
  id-mod-pkix1-algorithms(17) }
```

id-mgf1

FROM PKIX1-PSS-OAEP-Algorithms

```
{ iso(1) identified-organization(3) dod(6)
  internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
  id-mod-pkix1-rsa-pkalgs(33) }
```

AlgorithmIdentifier

FROM PKIX1Explicit88

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

;

--

-- We defined RSA keys from the modules [RFC3279](#) and [RFC4055](#)

--

RSAPublicKeyCapabilities ::= SEQUENCE {

```
  minKeySize      RSAKeySize,
  maxKeySize      RSAKeySize OPTIONAL
}
```

RSAKeySize ::= INTEGER (1024 | 2048 | 3072 | 7680 | 15360 |  
4096 | 8192, ...)

RsaSsa-Pss-sig-caps ::= SEQUENCE {

```
  hashAlg AlgorithmIdentifier,
  maskAlg AlgorithmIdentifier OPTIONAL,
  trailerField INTEGER DEFAULT 1
```



```
}

--
-- we define DH/DSA keys from the module RFC3279
--

DSAKeyCapabilities ::= CHOICE {
    keySizes          [0] SEQUENCE {
        minKeySize      DSAKeySize,
        maxKeySize      DSAKeySize OPTIONAL
    },
    keyParams          [1] pk-dsa.&Params
}

DSAKeySize ::= INTEGER (1024 | 2048 | 3072 | 7680 | 15360 )

--
-- we define Elliptical Curve keys from the module RFC3279
--

EC-SMimeCaps ::= SEQUENCE (SIZE (1..MAX)) OF ECParameters

END
```





### [Appendix C](#). Future Work

A future revision of [[RFC5912](#)] should be done at some point which expands the definition of the PUBLIC-KEY class and allows for an S/MIME Capability to be included in the class definition. This would encourage people to think about this as an issue when defining new public key structures in the future.

Author's Address

Jim Schaad  
Soaring Hawk Consulting  
  
Email: [ietf@augustcellars.com](mailto:ietf@augustcellars.com)