

NTP Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 28, 2016

D. Sibold
K. Teichel
PTB
S. Roettger
Google Inc.
R. Housley
Vigil Security
February 25, 2016

Protecting Network Time Security Messages with the Cryptographic Message
Syntax (CMS)

[draft-ietf-ntp-cms-for-nts-message-06](#)

Abstract

This document describes a convention for using the Cryptographic Message Syntax (CMS) to protect the messages in the Network Time Security (NTS) protocol. NTS provides authentication of time servers as well as integrity protection of time synchronization messages using Network Time Protocol (NTP) or Precision Time Protocol (PTP).

Requirements Language

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 [RFC 2119](#) [[RFC2119](#)].

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 August 28, 2016.

Internet-Draft

CMS4NTS

February 2016

Copyright Notice

Copyright (c) 2016 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

1.	Introduction	3
2.	CMS Conventions for NTS Message Protection	3
2.1.	Fields of the employed CMS Content Types	5
2.1.1.	ContentInfo	5
2.1.2.	SignedData	6
2.1.3.	EnvelopedData	8
3.	Implementation Notes: ASN.1 Structures and Use of the CMS	9
3.1.	Preliminaries	9
3.2.	Unicast Messages	9
3.2.1.	Access Messages	9
3.2.2.	Association Messages	10
3.2.3.	Cookie Messages	11
3.2.4.	Time Synchronization Messages	12
3.3.	Broadcast Messages	13
3.3.1.	Broadcast Parameter Messages	13
3.3.2.	Broadcast Time Synchronization Message	14
3.3.3.	Broadcast Keycheck	14
4.	Certificate Conventions	15
5.	IANA Considerations	16
5.1.	SMI Security for S/MIME Module Identifier Registry	16
5.2.	SMI Security for S/MIME CMS Content Type Registry	16
5.3.	SMI Security for PKIX Extended Key Purpose Registry	17
6.	Security Considerations	17
7.	Acknowledgements	17
8.	References	17
8.1.	Normative References	17

8.2. Informative References	18
Appendix A. ASN.1 Module	18
Authors' Addresses	18

[1. Introduction](#)

This document provides details on how to construct NTS messages in practice. NTS provides secure time synchronization with time servers using Network Time Protocol (NTP) [[RFC5905](#)] or Precision Time Protocol (PTP) [[IEEE1588](#)]. Among other things, this document describes a convention for using the Cryptographic Message Syntax (CMS) [[RFC5652](#)] to protect messages in the Network Time Security (NTS) protocol. Encryption is used to provide confidentiality of secrets, and digital signatures are used to provide authentication and integrity of content.

Sometimes CMS is used in an exclusively ASN.1 [[ASN1](#)] environment. In this case, the NTS message may use any syntax that facilitates easy implementation.

[2. CMS Conventions for NTS Message Protection](#)

Regarding the usage of CMS, we differentiate between three archetypes according to which the NTS message types can be structured. They are presented below. Note that the NTS Message Object that is at the core of each structure does not necessarily contain all the data needed for the particular message type, but may contain only that data which needs to be secured directly with cryptographic operations using the CMS. Specific information about what is included can be found in [Section 3](#).

NTS-Plain: This archetype is used for actual time synchronization messages (explicitly, the following message types: `time_request`, `time_response`, `server_broad`, see [[I-D.ietf-ntp-network-time-security](#)], Section 6) as well as for client-side messages of all unicast and broadcast bootstrapping exchanges (explicitly `client_assoc`, `client_cook` and `client_bpar`) as well as the broadcast keycheck exchange (`client_keycheck` and `server_keycheck`). This archetype does not make use of any CMS structures at all. Figure 1 illustrates this structure.



NTS-Encrypted-and-Signed: This archetype is used for secure transmission of the cookie (only for the server_cook message type,

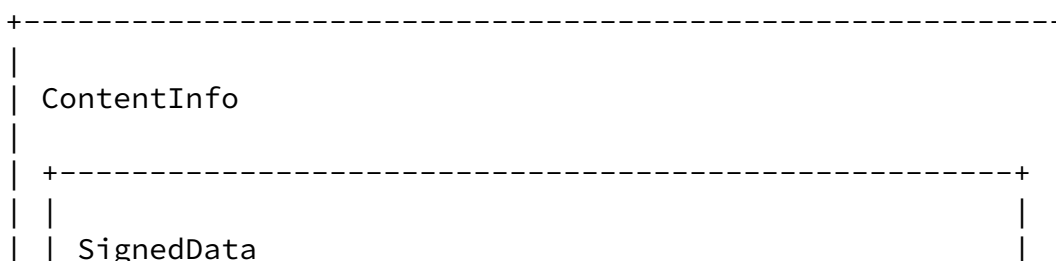
see [[I-D.ietf-ntp-network-time-security](#)], Section 6). For this, the following CMS structure is used:

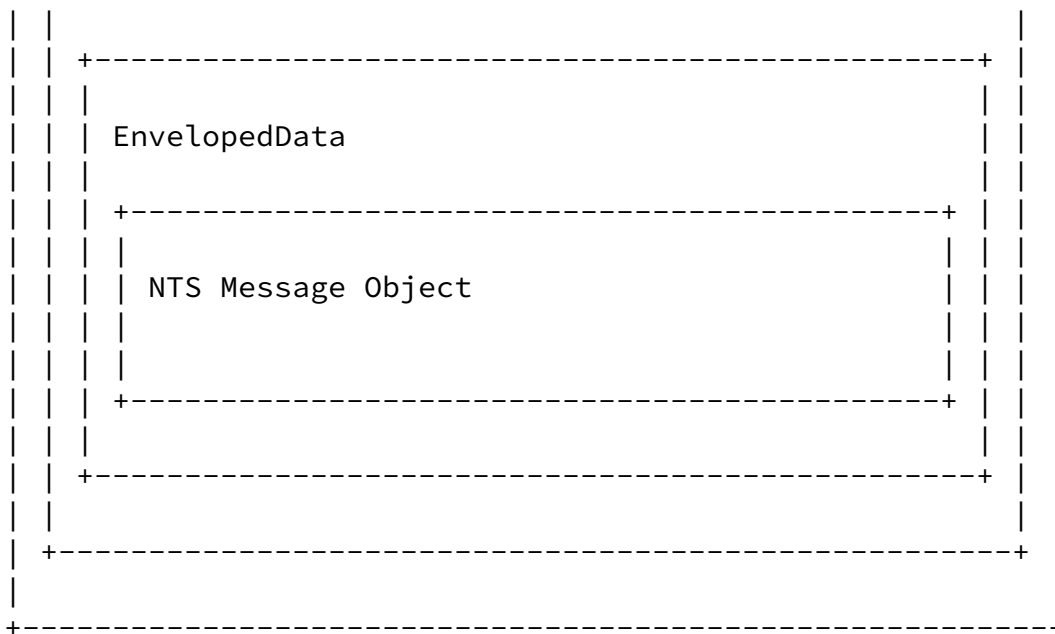
First, the NTS message MUST be encrypted using the EnvelopedData content type. EnvelopedData supports nearly any form of key management. In the NTS protocol the client provides a certificate in an unprotected message, and the public key from this certificate, if it is valid, will be used to establish a pairwise symmetric key for the encryption of the protected NTS message.

Second, the EnvelopedData content MUST be digitally signed using the SignedData content type. SignedData supports nearly any form of digital signature, and in the NTS protocol the server will include its certificate within the SignedData content type.

Third, the SignedData content type MUST be encapsulated in a ContentInfo content type.

Figure 2 illustrates this structure.



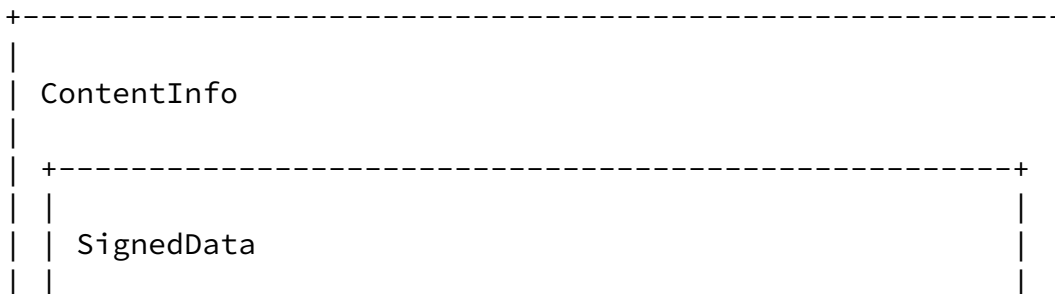


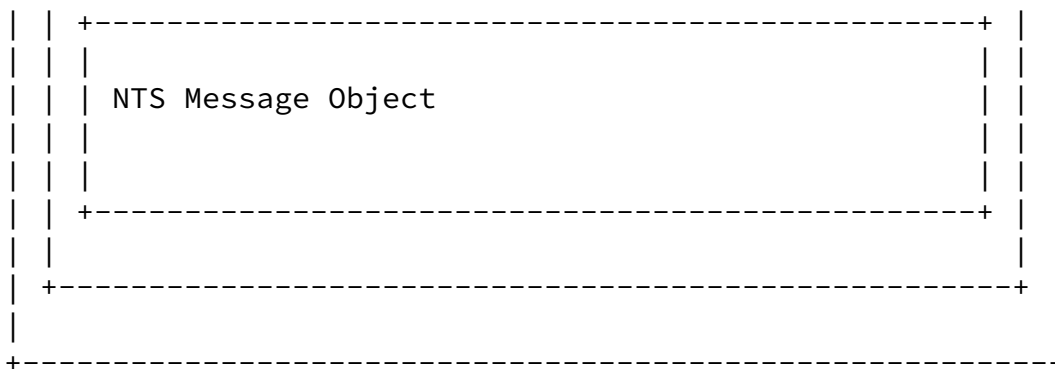
NTS-Signed: This archetype is used for `server_assoc` and `server_bpar` message types. It uses the following CMS structure:

First, the NTS message object MUST be wrapped in a `SignedData` content type. The messages MUST be digitally signed, and certificates included. `SignedData` supports nearly any form of digital signature, and in the NTS protocol the server will include its certificate within the `SignedData` content type.

Second, the `SignedData` content type MUST be encapsulated in a `ContentInfo` content type.

Figure 3 illustrates this structure.





[2.1.](#) Fields of the employed CMS Content Types

Overall, three CMS content types are used for NTS messages by the archetypes above. Explicitly, those content types are ContentInfo, SignedData and EnvelopedData. The following is a description of how the fields of those content types are used in detail.

[2.1.1.](#) ContentInfo

The ContentInfo content type is used in all archetypes except NTS-Plain. The fields of the ContentInfo content type are used as follows:

contentType -- indicates the type of the associated content. For all archetypes which use ContentInfo (these are NTS-Signed and

NTS-Encrypted-and-Signed), it MUST contain the object identifier for the SignedData content type:

```
id-signedData OBJECT IDENTIFIER ::= { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs7(7) 2 }
```

content -- is the associated content. For all archetypes using ContentInfo, it MUST contain the DER encoded SignedData content type.

[2.1.2.](#) SignedData

The SignedData content type is used in the NTS-Signed and NTS-Encrypted-and-Signed archetypes, but not in the NTS-Plain archetype. The fields of the SignedData content type are used as follows:

version -- the appropriate value depends on the optional items that are included. In the NTS protocol, the signer certificate MUST be included and other items MAY be included. The instructions in [\[RFC5652\] Section 5.1](#) MUST be followed to set the correct value.

digestAlgorithms -- is a collection of message digest algorithm identifiers. In the NTS protocol, there MUST be exactly one algorithm identifier present. The instructions in [Section 5.4 of \[RFC5652\]](#) MUST be followed.

encapContentInfo -- this structure is always present. In the NTS protocol, it MUST follow these conventions:

eContentType -- is an object identifier. In the NTS protocol, for the NTS-Signed archetype, it MUST identify the type of the NTS message that was encapsulated. For the NTS-Encrypted-and-Signed archetype, it MUST contain the object identifier for the EnvelopedData content type:

```
id-envelopedData OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs7(7) 3 }.
```

eContent is the content itself, carried as an octet string. For the NTS-Signed archetype, it MUST contain the DER encoded encapsulated NTS message object. The instructions in [Section 6.3 of \[RFC5652\]](#) MUST be followed. For the NTS-Encrypted-and-Signed archetype, it MUST contain the DER encoded EnvelopedData content type.

certificates -- is a collection of certificates. In the NTS protocol, it MUST contain the DER encoded certificate [\[RFC5280\]](#) of

the sender. It is intended that the collection of certificates be sufficient for the recipient to construct a certification path from a recognized "root" or "top-level certification authority" to the certificate used by the sender.

crls -- is a collection of revocation status information. In the NTS protocol, it MAY contain one or more DER encoded CRLs [\[RFC5280\]](#). It is intended that the collection contain information

sufficient to determine whether the certificates in the certificates field are valid.

signerInfos -- is a collection of per-signer information. In the NTS protocol, for the NTS-Signed and the NTS-Encrypted-and-Signed archetypes, there MUST be exactly one SignerInfo structure present. The details of the SignerInfo type are discussed in [Section 5.3 of \[RFC5652\]](#). In the NTS protocol, it MUST follow these conventions:

version -- is the syntax version number. In the NTS protocol, the SignerIdentifier is subjectKeyIdentifier, therefore the version MUST be 3.

sid -- identifies the signer's certificate. In the NTS protocol, the "sid" field contains the subjectKeyIdentifier from the signer's certificate.

digestAlgorithm -- identifies the message digest algorithm and any associated parameters used by the signer. In the NTS protocol, the identifier MUST match the single algorithm identifier present in the digestAlgorithms.

signedAttrs -- is a collection of attributes that are signed. In the NTS protocol, it MUST be present, and it MUST contain the following attributes:

Content Type -- see [Section 11.1 of \[RFC5652\]](#).

Message Digest -- see [Section 11.2 of \[RFC5652\]](#).

In addition, it MAY contain the following attributes:

Signing Time -- see [Section 11.3 of \[RFC5652\]](#).

Binary Signing Time -- see [Section 3 of \[RFC5652\]](#).

signatureAlgorithm -- identifies the signature algorithm and any associated parameters used by the signer to generate the digital signature.

signature is the result of digital signature generation using

the message digest and the signer's private key. The instructions in [Section 5.5 of \[RFC5652\]](#) MUST be followed.

unsignedAttrs -- is an optional collection of attributes that are not signed. In the NTS protocol, it MUST be absent.

[2.1.3.](#) EnvelopedData

The EnvelopedData content type is used only in the NTS-Encrypted-and-Signed archetype. The fields of the EnvelopedData content type are used as follows:

version -- the appropriate value depends on the type of key management that is used. The instructions in [\[RFC5652\]](#) [Section 6.1](#) MUST be followed to set the correct value.

originatorInfo -- this structure is present only if required by the key management algorithm. In the NTS protocol, it MUST be present when a key agreement algorithm is used, and it MUST be absent when a key transport algorithm is used. The instructions in [Section 6.1 of \[RFC5652\]](#) MUST be followed.

recipientInfos -- this structure is always present. In the NTS protocol, it MUST contain exactly one entry that allows the client to determine the key used to encrypt the NTS message. The instructions in [Section 6.2 of \[RFC5652\]](#) MUST be followed.

encryptedContentInfo -- this structure is always present. In the NTS protocol, it MUST follow these conventions:

contentType -- indicates the type of content. In the NTS protocol, it MUST identify the type of the NTS message that was encrypted.

contentEncryptionAlgorithm -- identifies the content-encryption algorithm and any associated parameters used to encrypt the content.

encryptedContent -- is the encrypted content. In the NTS protocol, it MUST contain the encrypted NTS message. The instructions in [Section 6.3 of \[RFC5652\]](#) MUST be followed.

unprotectedAttrs -- this structure is optional. In the NTS protocol, it MUST be absent.

[3.](#) Implementation Notes: ASN.1 Structures and Use of the CMS

This section presents some hints about the structures of the NTS message objects for the different message types when one wishes to implement the security mechanisms.

[3.1.](#) Preliminaries

The following ASN.1 coded data types "NTSAccessKey", "NTSNonce", and "NTSVersion" are needed for other types used below for NTS messages. "NTSAccessKey" specifies an access key, which is required for limitation of client association requests.

```
NTSAccessKey ::= OCTET STRING (SIZE(16))
```

"NTSNonce" specifies a 128 bit nonce as required in several message types.

```
NTSNonce ::= OCTET STRING (SIZE(16))
```

"NTSVersion" specifies a version number, which is required for version negotiation.

```
NTSVersion ::= INTEGER (0..255)
```

The following ASN.1 coded data types are also necessary for other types.

```
KeyEncryptionAlgorithmIdentifiers ::=  
    SET OF KeyEncryptionAlgorithmIdentifier
```

```
ContentEncryptionAlgorithmIdentifiers ::=  
    SET OF ContentEncryptionAlgorithmIdentifier
```

[3.2.](#) Unicast Messages

[3.2.1.](#) Access Messages

[3.2.1.1.](#) Message Type: "client_access"

This message is structured according to the NTS-Plain archetype. There is no data necessary besides that which is transported in the NTS message object, which is an ASN.1 object of type "ClientAccessData" and structured as follows:

```
ClientAccessData ::= NULL
```

It is identified by the following object identifier:

id-ct-nts-clientAccess OBJECT IDENTIFIER ::= TBD1

[3.2.1.2](#). Message Type: "server_access"

This message is structured according to the NTS-Plain archetype. There is no data necessary besides that which is transported in the NTS message object, which is an ASN.1 object of type "ServerAccessData" and structured as follows:

```
ServerAccessData ::= SEQUENCE {  
    accessKey          NTSAccessKey  
}
```

It is identified by the following object identifier:

id-ct-nts-serverAccess OBJECT IDENTIFIER ::= TBD2

[3.2.2](#). Association Messages

[3.2.2.1](#). Message Type: "client_assoc"

This message is structured according to the NTS-Plain archetype. There is no data necessary besides that which is transported in the NTS message object, which is an ASN.1 object of type "ClientAssocData" and structured as follows:

```
ClientAssocData ::= SEQUENCE {  
    accessKey          NTSAccessKey,  
    nonce              NTSNonce,  
    minVersion         NTSVersion,  
    hmacHashAlgos     DigestAlgorithmIdentifiers,  
    keyEncAlgos        KeyEncryptionAlgorithmIdentifiers,  
    contentEncAlgos    ContentEncryptionAlgorithmIdentifiers  
}
```

It is identified by the following object identifier:

id-ct-nts-clientAssoc OBJECT IDENTIFIER ::= TBD3

[3.2.2.2.](#) Message Type: "server_assoc"

This message is structured according to the NTS-Signed archetype. It requires additional data besides that which is transported in the NTS message object, namely the signature and certificate chain are included in the appropriate fields of the "SignedData" CMS structure that the NTS message object is wrapped in. The NTS message object itself is an ASN.1 object of type "ServerAssocData" and structured as follows:

```
ServerAssocData ::= SEQUENCE {
    nonce                NTSNonce,
    proposedVersion     NTSVersion,
    hmacHashAlgos       DigestAlgorithmIdentifiers,
    choiceHmacHashAlgo  DigestAlgorithmIdentifier,
    keyEncAlgos         KeyEncryptionAlgorithmIdentifiers,
    choiceKeyEncAlgo    KeyEncryptionAlgorithmIdentifier,
    contentEncAlgos     ContentEncryptionAlgorithmIdentifiers,
    choiceContentEncAlgo ContentEncryptionAlgorithmIdentifier
}
```

It is identified by the following object identifier:

```
id-ct-nts-serverAssoc OBJECT IDENTIFIER ::= TBD4
```

[3.2.3.](#) Cookie Messages

[3.2.3.1.](#) Message Type: "client_cook"

This message is structured according to the NTS-Plain archetype. It requires no additional data besides that which is transported in the NTS message object. The NTS message object itself is an ASN.1 object of type "ClientCookieData" and structured as follows:

```
ClientCookieData ::= SEQUENCE {
    nonce                NTSNonce,
    signAlgo            SignatureAlgorithmIdentifier,
    hmacHashAlgo        DigestAlgorithmIdentifier,
    encAlgo             ContentEncryptionAlgorithmIdentifier,
    keyEncAlgo          KeyEncryptionAlgorithmIdentifier,
    certificates         CertificateSet
}
```

It is identified by the following object identifier:

id-ct-nts-clientCookie OBJECT IDENTIFIER ::= TBD5

[3.2.3.2.](#) Message Type: "server_cook"

This message is structured according to the "NTS-Encrypted-and-Signed" archetype. It requires additional data besides that which is transported in the NTS message object, namely the signature is included in the appropriate field of the "SignedData" CMS structure that the NTS message object is wrapped in. The NTS message object itself is an ASN.1 sequence of type "ServerCookieData" and structured as follows:

```
ServerCookieData ::= SEQUENCE {  
    nonce      NTSNonce,  
    cookie     OCTET STRING (SIZE(16))  
}
```

It is identified by the following object identifier:

id-ct-nts-serverCookie OBJECT IDENTIFIER ::= TBD6

[3.2.4.](#) Time Synchronization Messages

[3.2.4.1.](#) Message Type: "time_request"

This message is structured according to the "NTS-Plain" archetype.

This message type requires additional data to that which is included in the NTS message object, namely it requires regular time synchronization data, as an unsecured packet from a client to a server would contain. Optionally, it requires the Message Authentication Code (MAC) to be generated over the whole rest of the packet (including the NTS message object) and transported in some way. The NTS message object itself is an ASN.1 object of type "TimeRequestSecurityData", whose structure is as follows:

```
TimeRequestSecurityData ::=
```

```
SEQUENCE {
    nonce          NTSNonce,
    hmacHashAlgo  DigestAlgorithmIdentifier,
    keyInputValue OCTET STRING (SIZE(16))
}
```

It is identified by the following object identifier:

id-ct-nts-securityDataReq OBJECT IDENTIFIER ::= TBD7

[3.2.4.2.](#) Message Type: "time_response"

This message is structured according to the "NTS-Plain" archetype.

It requires two items of data in addition to that which is transported in the NTS message object. Like "time_request", it requires regular time synchronization data. Furthermore, it requires the Message Authentication Code (MAC) to be generated over the whole rest of the packet (including the NTS message object) and transported in some way. The NTS message object itself is an ASN.1 object of type "TimeResponseSecurityData", with the following structure:

```
TimeResponseSecurityData ::=
SEQUENCE {
    nonce    NTSNonce,
}
```

It is identified by the following object identifier:

id-ct-nts-securityDataResp OBJECT IDENTIFIER ::= TBD8

[3.3.](#) Broadcast Messages

[3.3.1.](#) Broadcast Parameter Messages

[3.3.1.1.](#) Message Type: "client_bpar"

This first broadcast message is structured according to the NTS-Plain archetype. There is no data necessary besides that which is transported in the NTS message object, which is an ASN.1 object of

type "BroadcastParameterRequest" and structured as follows:

```
BroadcastParameterRequest ::=
SEQUENCE {
    nonce      NTSNonce,
    clientId   SubjectKeyIdentifier
}
```

It is identified by the following object identifier:

```
id-ct-nts-broadcastParamReq OBJECT IDENTIFIER ::= TBD9
```

[3.3.1.2](#). Message Type: "server_bpar"

This message is structured according to "NTS-Signed". It requires additional data besides that which is transported in the NTS message object, namely the signature is included in the appropriate field of the "SignedData" CMS structure that the NTS message object is wrapped in. The NTS message object itself is an ASN.1 object of type "BroadcastParameterResponse" and structured as follows:

```
BroadcastParameterResponse ::=
SEQUENCE {
    nonce                NTSNonce,
    oneWayAlgo1         DigestAlgorithmIdentifier,
    oneWayAlgo2         DigestAlgorithmIdentifier,
    lastKey             OCTET STRING (SIZE (16)),
    intervalDuration    BIT STRING,
    disclosureDelay     INTEGER,
    nextIntervalTime    BIT STRING,
    nextIntervalIndex   INTEGER
}
```

It is identified by the following object identifier:

id-ct-nts-broadcastParamResp OBJECT IDENTIFIER ::= TBD10

[3.3.2.](#) Broadcast Time Synchronization Message

[3.3.2.1.](#) Message Type: "server_broad"

This message is structured according to the "NTS-Plain" archetype. It requires regular broadcast time synchronization data in addition to that which is carried in the NTS message object. Like "time_response", this message type also requires a MAC, generated over all other data, to be transported within the packet. The NTS message object itself is an ASN.1 object of type "BroadcastTime". It has the following structure:

```
BroadcastTime ::=
SEQUENCE {
    thisIntervalIndex    INTEGER,
    disclosedKey         OCTET STRING (SIZE (16)),
}
```

It is identified by the following object identifier:

id-ct-nts-broadcastTime OBJECT IDENTIFIER ::= TBD11

[3.3.3.](#) Broadcast Keycheck

[3.3.3.1.](#) Message Type: "client_keycheck"

This message is structured according to the "NTS-Plain" archetype. There is no data necessary besides that which is transported in the NTS message object, which is an ASN.1 object of type "ClientKeyCheckSecurityData" and structured as follows:

```
ClientKeyCheckSecurityData ::=
SEQUENCE {
    nonce_k            NTSNonce,
    interval_number    INTEGER,
```



```
    hmacHashAlgo      DigestAlgorithmIdentifier,
    keyInputValue     OCTET STRING (SIZE(16))
}
```

It is identified by the following object identifier:

id-ct-nts-clientKeyCheck OBJECT IDENTIFIER ::= TBD12

3.3.3.2. Message Type: "server_keycheck"

This message is also structured according to "NTS-Plain". It requires only a MAC, generated over the NTS message object, to be included in the packet in addition to what the NTS message object itself contains. The latter is an ASN.1 object of type "ServerKeyCheckSecurityData", which is structured as follows:

```
ServerKeyCheckSecurityData ::=
SEQUENCE {
    nonce                NTSNonce,
    interval_number     INTEGER
}
```

It is identified by the following object identifier:

id-ct-nts-serverKeyCheck OBJECT IDENTIFIER ::= TBD13

4. Certificate Conventions

The syntax and processing rules for certificates are specified in [\[RFC5280\]](#). In the NTS protocol, the server certificate MUST contain the following extensions:

Subject Key Identifier -- see [Section 4.2.1.2 of \[RFC5280\]](#).

Key Usage -- see [Section 4.2.1.3 of \[RFC5280\]](#).

Extended Key Usage -- see [Section 4.2.1.12 of \[RFC5280\]](#).

For a certificate issued to a time server, the Extended Key Usage extension MAY include the id-kp-ntsServerAuth object identifier. When a certificate issuer includes this object identifier in the extended key usage extension, it provides an attestation that the certificate subject is a time server that supports the NTS protocol. The extension MAY also include the id-kp-ntsServerAuthz object

identifier. When a certificate issuer includes this object identifier in the extended key usage extension, it provides an attestation that the certificate subject is a time server which the issuer believes to be willing and able to disseminate correct time (for example, this can be used to signal a server's authorization to disseminate legal time).

For a certificate issued to a time client, the Extended Key Usage extension MAY include the `id-kp-ntsClientAuthz` object identifier. When a certificate issuer includes this object identifier in the extended key usage extension, it provides an attestation that the certificate subject is a time client who has authorization from the issuer to access secured time synchronization (for example, this can be used to provide access in the case of a paid service for secure time synchronization).

[5.](#) IANA Considerations

[5.1.](#) SMI Security for S/MIME Module Identifier Registry

Within the "SMI Security for S/MIME Module Identifier (1.2.840.113549.1.9.16.0)" table, add one module identifier:

Decimal	Description	References
TBD0	<code>id-networkTimeSecurity-2015</code>	[this doc]

[5.2.](#) SMI Security for S/MIME CMS Content Type Registry

Within the "SMI Security for S/MIME CMS Content Type (1.2.840.113549.1.9.16.1)" table, add thirteen content type identifiers:

Decimal	Description	References
TBD1	<code>id-ct-nts-clientAccess</code>	[this doc]
TBD2	<code>id-ct-nts-serverAccess</code>	[this doc]
TBD3	<code>id-ct-nts-clientAssoc</code>	[this doc]
TBD4	<code>id-ct-nts-serverAssoc</code>	[this doc]
TBD5	<code>id-ct-nts-clientCookie</code>	[this doc]
TBD6	<code>id-ct-nts-serverCookie</code>	[this doc]
TBD7	<code>id-ct-nts-securityDataReq</code>	[this doc]
TBD8	<code>id-ct-nts-securityDataResp</code>	[this doc]
TBD9	<code>id-ct-nts-broadcastParamReq</code>	[this doc]
TBD10	<code>id-ct-nts-broadcastParamResp</code>	[this doc]
TBD11	<code>id-ct-nts-broadcastTime</code>	[this doc]
TBD12	<code>id-ct-nts-clientKeyCheck</code>	[this doc]

[5.3.](#) SMI Security for PKIX Extended Key Purpose Registry

Within the "SMI Security for PKIX Extended Key Purpose Identifiers (1.3.6.1.5.5.7.3)" table, add three key purpose identifiers:

Decimal	Description	References
TBD14	id-kp-ntsServerAuth	[this doc]
TBD15	id-kp-ntsServerAuthz	[this doc]
TBD16	id-kp-ntsClientAuthz	[this doc]

[6.](#) Security Considerations

For authentication the server's certificate MAY contain an extended key purpose identifier (id-kp-ntsServerAuth). Additionally the identifiers id-kp-ntsServerAuthz and id-kp-ntsClientAuthz MAY be used to grant the associated roles to the certified entity in the time dissemination infrastructure (see also [Appendix D](#) in [\[I-D.ietf-ntp-network-time-security\]](#)).

[7.](#) Acknowledgements

The authors would like to thank Harlan Stenn, Richard Welty and Martin Langer for their technical review and specific text contributions to this document.

[8.](#) References

[8.1.](#) Normative References

- [ASN1] International Telecommunication Union, "Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, November 2008.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [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](#), DOI 10.17487/RFC5280, May 2008, <<http://www.rfc-editor.org/info/rfc5280>>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, [RFC 5652](#), DOI 10.17487/RFC5652, September 2009, <<http://www.rfc-editor.org/info/rfc5652>>.

Sibold, et al.

Expires August 28, 2016

[Page 17]

Internet-Draft

CMS4NTS

February 2016

[8.2.](#) Informative References

[I-D.ietf-ntp-network-time-security]

Sibold, D., Roettger, S., and K. Teichel, "Network Time Security", [draft-ietf-ntp-network-time-security-13](#) (work in progress), February 2016.

[IEEE1588]

IEEE Instrumentation and Measurement Society. TC-9 Sensor Technology, "IEEE standard for a precision clock synchronization protocol for networked measurement and control systems", 2008.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", [RFC 5905](#), DOI 10.17487/RFC5905, June 2010, <<http://www.rfc-editor.org/info/rfc5905>>.

[Appendix A.](#) ASN.1 Module

The ASN.1 module contained in this appendix defines the id-kp-NTSserver object identifier.

```
NTSserverKeyPurpose
  { TBD }
```

```
DEFINITIONS IMPLICIT TAGS ::=
BEGIN
```

```
id-kp-NTSserver OBJECT IDENTIFIER ::= { TBD17 }
```

```
END
```

Authors' Addresses

Dieter Sibold
Physikalisch-Technische Bundesanstalt
Bundesallee 100
Braunschweig D-38116
Germany

Phone: +49-(0)531-592-8420
Fax: +49-531-592-698420
Email: dieter.sibold@ptb.de

Sibold, et al.

Expires August 28, 2016

[Page 18]

Internet-Draft

CMS4NTS

February 2016

Kristof Teichel
Physikalisch-Technische Bundesanstalt
Bundesallee 100
Braunschweig D-38116
Germany

Phone: +49-(0)531-592-8421
Email: kristof.teichel@ptb.de

Stephen Roettger
Google Inc.

Email: stephen.roettger@googlemail.com

Russ Housley
Vigil Security
918 Spring Knoll Drive
Herndon, VA 20170

Email: housley@vigilsec.com

