NTP Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 26, 2015

D. Sibold
PTB
S. Roettger
Google Inc
K. Teichel
PTB
R. Housley
Vigil Security
October 23, 2014

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

draft-ietf-ntp-cms-for-nts-message-00.txt

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 April 26, 2015.

Table of Contents

1. Introduction

This document provides detail 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 four archetypes according to which the NTS message types can be structured:

NTS-Plain: This archetype is used for actual time synchronization messages (explicitly, the message types: time_request, time_response, server_broad, see [I-D.ietf-ntp-network-time-security], section 6) as well as for the very first messages of a unicast or a broadcast exchange (client_assoc or client_bpar, respectively) and the broadcast keycheck exchange (client_keycheck and server_keycheck). This archetype does not make use of any CMS structures.

NTS-Signed-and-Encrypted: 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.

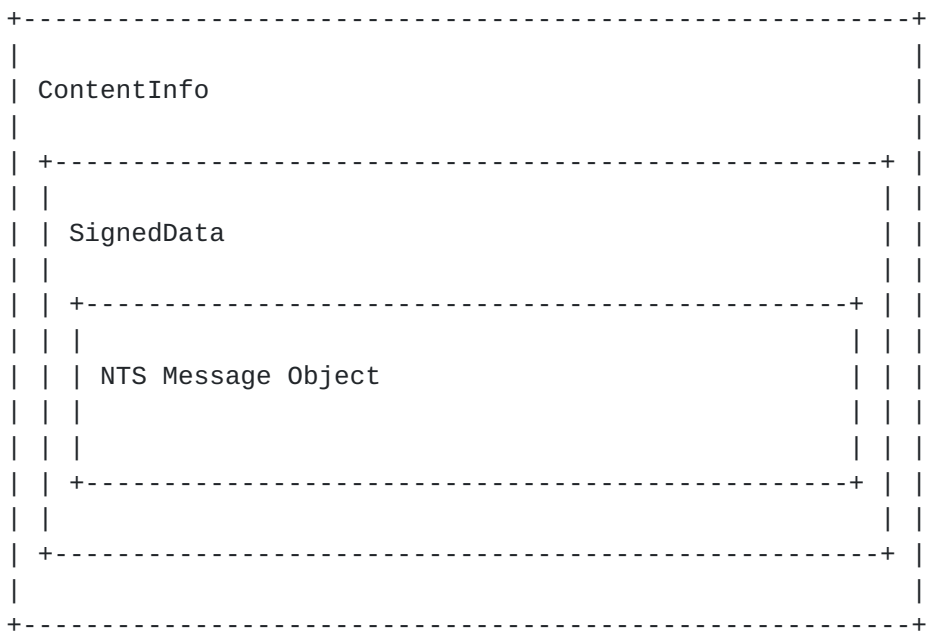
Figure 1 illustrates this structure.

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 2 illustrates this structure.



NTS-Certified: This archetype is used for the client_cook message type. It uses a CMS structure much like the NTS-Signed archetype (see Figure 2), with the only difference being that messages SHOULD NOT be digitally signed. This archetype employs the CMS structure merely in order to transport certificates.

Whichever archetype is used, the resulting structure is always transported in an extension field of an NTP packet. In the case of messages that also need to carry time synchronization data, this data is written into the regular fields of the NTP packet.

2.1. Fields of the employed CMS Content Types

Overall, three CMS content types are used for NTS messages: 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 four archetypes. The fields of the SignedData content type are used as follows:

contentType -- indicates the type of the associated content. For the archetype NTS-Plain, it MUST identify the NTS message object that is included. For all other archetypes (NTS-Certified, NTS-Signed and NTS-Signed-and-Encrypted), 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 the NTS-Plain archetype, it MUST contain the DER encoded NTS message object. For all other archetypes, it MUST contain the DER encoded SignedData content type.

2.1.2. SignedData

The SignedData content type is used in the NTS-Certified, NTS-Signed and NTS-Signed-and-Encrypted 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-Certified and NTS-Signed archetypes, it MUST identify the type of the NTS message that was encapsulated. For the NTS-Signed-and-Encrypted 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-Certified and NTS-Signed archetypes, it MUST contain the DER encoded encapsulated NTS message object. The instructions in Section 6.3 of [RFC5652] MUST be followed. For the NTS-Signed-and-Encrypted 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-Certified archetype, this SHOULD be left out. For both the NTS-Signed and the NTS-Signed-and-Encrypted 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, the it MUST be absent.

2.1.3. EnvelopedData

The EnvelopedData content type is used only in the NTS-Signed-and-Encrypted 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 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 the 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. Certificate Conventions

The syntax and processing rules for certificates are specified in [RFC5652]. In the NTS protocol, the server certificate MUST contain the following extensions:

Subject Key Identifier -- see Section 4.2.1.2 of [RFC5652].

Key Usage -- see Section 4.2.1.3 of [RFC5652].

Extended Key Usage -- see Section 4.2.1.22 of [RFC5652].

The Extended Key Usage extension MUST include the id-kp-NTSserver 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 id-kp-NTSserver object identifier is:

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

4. Implementation Notes: ASN.1 Structures and Use of the CMS

This section gives some hints on the structures of the NTS message objects for the different message types when one wishes to implement the protocol.

4.1. Preliminaries

The following ASN.1 coded data type "NTSNonce" is needed for other types used below for NTS messages. It specifies a 128 bit nonce as required in several message types:

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

4.2. Unicast Messages

4.2.1. Association Messages

4.2.1.1. Message Type: "client_assoc"

This message is structured according to the NTS-Plain archetype. It is realized as an NTP packet with an extension field which holds all the data relevant for NTS. Explicitly, the extension field contains an ASN.1 object of type "ClientAssocData", which is structured as follows:

```
ClientAssocData ::= SEQUENCE {
    clientId         SubjectKeyIdentifier,
    digestAlgos      DigestAlgorithmIdentifiers,
    keyEncAlgos      KeyEncryptionAlgorithms,
    contentEncAlgos  ContentEncryptionAlgorithms
}
```

4.2.1.2. Message Type: "server_assoc"

This message is structured according to the NTS-Signed archetype. The NTS message object in this case is an ASN.1 object of type "ServerAssocData", which is structured as follows:

```
ServerAssocData ::= SEQUENCE {
    clientId              SubjectKeyIdentifier,
    choiceDigestAlgo      DigestAlgorithmIdentifier,
    choiceKeyEncAlgo      KeyEncryptionAlgorithmIdentifier,
    choiceContentEncAlgo  ContentEncryptionAlgorithmIdentifier
}
```

4.2.2. Cookie Messages

4.2.2.1. Message Type: "client_cook"

This message is structured according to the NTS-Certified archetype. The NTS message object is a "ClientCookieData" type ASN.1 object, structured as follows:

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

It is identified by the following object identifier (fictional values):

```
id-clientCookieData OBJECT IDENTIFIER ::=
    {nts(??) cookie(3) clientcookiedata(1)}
```

4.2.2.2. Message Type: "server_cook"

This message is structured according to the "NTS-Signed-and-Encrypted" archetype. The NTS message object is a "ServerCookieData" object, specified as:

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

It is identified by the following object identifier (fictional values):

```
id-serverCookieData OBJECT IDENTIFIER ::=
    {nts(??) cookie(3) servercookiedata(2)}
```

4.2.3. Time Synchronization Messages

4.2.3.1. Message Type: "time_request"

This message is structured according to the "NTS-Plain" archetype. It is realized as an NTP packet which actually contains regular NTP time synchronization data, as an unsecured NTP packet from a client to a server would. Furthermore, the packet has an extension field which contains an ASN.1 object of type "TimeRequestSecurityData", whose structure is as follows:

```
TimeRequestSecurityData ::=
SEQUENCE {
    nonce_t           NTSNonce,
    digestAlgo        DigestAlgorithmIdentifier,
    hashOfClientCert  BIT STRING
}
```

4.2.3.2. Message Type: "time_response"

This message is also structured according to "NTS-Plain". It is realized as an NTP packet which, like "time_request", contains regular NTP time synchronization data, as an unsecured NTP packet from a server back to a client would. The packet also has an extension field which contains an ASN.1 object of type "TimeResponseSecurityData", with the following structure:

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

Finally, this NTP packet has a MAC field which contains a Message Authentication Code generated over the whole packet (including the extension field).

4.3. Broadcast Messages

4.3.1. Broadcast Parameter Messages

4.3.1.1. Message Type: "client_bpar"

This first broadcast message is structured according to the NTS-Plain archetype. It is realized as an NTP packet which is empty except for an extension field which contains ans ASN.1 object of type "BroadcastParameterRequest", which is structured as follows:

```
BroadcastParameterRequest ::=
SEQUENCE {
    clientId  SubjectKeyIdentifier
}
```

4.3.1.2. Message Type: "server_bpar"

This message is structured according to "NTS-Signed". It is realized as an NTP packet whose extension field carries the necessary CMS structure. The NTS message object in this case is an ASN.1 object of type "BroadcastParameterResponse", with the following structure:

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

4.3.2. Broadcast Time Synchronization Message

4.3.2.1. Message Type: "server_broad"

This message is structured according to the "NTS-Plain" archetype. Its realization works via an NTP packet which carries regular NTP broadcast time data as well as an extension field, which contains an ASN.1 object of type "BroadcastTime". It has the following structure:

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

In addition, this packet has a MAC field which contains a Message Authentication Code generated over the whole packet (including the extension field).

4.3.3. Broadcast Keycheck

4.3.3.1. Message Type: "client_keycheck"

This message is structured according to the "NTS-Plain" archetype. It is realized as an NTP packet with an extension field, which contains an ASN.1 object of type "ClientKeyCheckSecurityData", whose structure is as follows:

```
ClientKeyCheckSecurityData ::=
SEQUENCE {
    nonce_k           NTSNonce,
    interval_number   INTEGER,
    digestAlgo        DigestAlgorithmIdentifier,
    hashOfClientCert  BIT STRING
}
```

4.3.3.2. Message Type: "server_keycheck"

This message is also structured according to "NTS-Plain". It is also realized as an NTP packet with an extension field, which contains an ASN.1 object of type "ServerKeyCheckSecurityData", with the following structure:

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

Additionally, this NTP packet has a MAC field which contains a Message Authentication Code generated over the whole packet (including the extension field).

5. IANA Considerations

IANA needs to assign an object identifier for id-kp-NTSserver key purpose and another one for the ASN.1 module in the appendix.

6. Security Considerations

To be written.

7. References

7.1. Normative References

[ASN1] International Telecommunication Union, "Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, November 2008.

[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.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.

[RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

7.2. 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-04 (work in progress), July 2014.

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 ::= { TBD }

      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

Stephen Roettger
Google Inc

Email: stephen.roettger@googlemail.com

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

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

Russ Housley
Vigil Security