Workgroup: NTP WG
Published: 4 June 2021
Intended Status: Standards Track
Expires: 6 December 2021
Authors: H. Gerstung   M. Rohde   D. Arnold
Meinberg   Meinberg   Meinberg

Network Time Security for the Unicast Mode of the Precision Time Protocol

Abstract

This memo specifies the application of Network Time Security, a mechanism for using Transport Layer Security (TLS) and Authenticated Encryption with Associated Data (AEAD) to provide cryptographic security for the unicast mode of the Precision Time Protocol.

It is based on the 'Network Time Security for the Network Time Protocol' document RFC8915 and re-uses most of its mechanisms for providing a secure and robust key exchange solution for unicast PTP. Due to the different modes of operation, additional steps are required to secure unicast PTP communication between the PTP clients and unicast PTP servers. In addition to defining the new record types and other required values to allow the utilization of the NTS key exchange sub protocol, there are a number of additional protocol enhancements and server-side requirements which are defined in this memo.

NOTE

This document is work in progress

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This memo specifies Network Time Security for unicast mode of the Precision Time Protocol (PTP). It is based on [RFC8915] and applies the key exchange mechanism described there to PTP. The Precision Time Protocol is standardized in [IEEE1588] and offers a number of different modes and mappings to communication protocols. The security mechanisms described here provide a way to secure the unicast mode of PTP as specified in sub clause 16.1 of [IEEE1588].

The PTP integrated security mechanism has been specified in sub clause 16.14 of [IEEE1588] and introduces an AUTHENTICATION TLV that carries all necessary information to enable the receiver of a PTP message to verify its integrity. Although two different approaches are described in that sub clause (immediate and delayed security processing), NTS4UPTP only uses the immediate security processing.

In addition to sub clause 16.14, Annex P of [IEEE1588] provides additional explanation and description of PTP security. It is stated there that for key management it is assumed that a separate mechanism outside the context of PTP is used. In P.2.1.2 the document clearly states that this assumption was made in relation to the security mechanism described in 16.14 and it goes on to discuss some Key management options and it names both manual and automatic key management as possible approaches.

This memo describes a way to use the automatic key exchange mechanism as defined in [RFC8915] as the key management for unicast PTP. The NTS-KE protocol has clearly been designed to support using it for multiple time synchronization protocols and this document is utilizing this support.

1.1. Terminology

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. Objectives

The objectives of NTS are defined in [RFC8915] and, with some exceptions, apply to NTS4UPTP as well.

* Identity: Through the use of a X.509 public key infrastructure, implementations can cryptographically establish the identity of the parties they are communicating with.

* Authentication: Implementations can cryptographically verify that any time synchronization packets are authentic, i.e., that they were produced by an identified party and have not been modified in transit.

* Replay prevention: clients and servers can detect when a received packet is a replay of a previous packet.

* Request-response consistency: clients can verify that a unicast PTP packet received from a server was sent in response to a particular request from the client.

* Non-amplification: Implementations (especially server implementations) can avoid acting as distributed denial-of-service (DDoS) amplifiers by never responding to a packet with one or more packets creating more traffic than the initiating packet.

* Scalability: Server implementations can serve large numbers of clients.

* Performance: NTS must not significantly degrade the quality of the time transfer. The encryption and authentication used when actually transferring time should be lightweight.

The following objectives of [RFC8915] are not met in this proposal:

* Confidentiality: Basic time synchronization data is considered nonconfidential and sent in the clear. Despite this, NTS4NTP includes support for encrypting NTP extension fields. NTS4UPTP does not offer this kind of support as it is not considered useful or required for unicast PTP implementations.

* Unlinkability: For mobile clients, NTS4NTP does not leak any information additional to NTP which would permit a passive adversary to determine that two packets sent over different networks came from the same client. This objection cannot be achieved by unicast PTP because the protocol requires the server to keep a state for all its clients. It is also not considered a requirement in most applications where unicast PTP is deployed.

3. Application of the NTS protocol to PTP

Unlike NTP [RFC5905] which uses a request-response communication approach, the unicast mode of PTP is applying a subscription based model. Although [IEEE1588] allows unicast operation without negotiation this is rarely used. For this reason only unicast PTP with negotiation is considered. A PTP client (PTP Ordinary Clock, or synchronizing port of a PTP unicast Boundary Clock) sends a request for the transmission of packets to a PTP instance offering such a service. This could be a PTP unicast Grandmaster instance or a PTP boundary clock. Note that [IEEE1588] allows PTP Ports in a states other than master to accept unicast message grant requests and act as a unicast PTP master. This option can be used for monitoring purposes. In this memo we treat all unicast associations in the same way regardless of whether it is for purposes of time transfer or monitoring, and refer to the port that grants message contracts as a PTP server.

A PTP server that receives a message grant request will then either accept the request or deny it, for example based on capacity considerations or its own operational state. This sub protocol of

IEEE 1588 is called unicast message negotiation, an optional feature defined in sub clause 16.1 of [IEEE1588]. Both the PTP client and the PTP server granting a request can cancel a subscription (referred to as contract in PTP) after it has been granted and each contract includes a duration after which the PTP instance stops sending packets automatically if the PTP client did not request a new contract before the old contract ended.

This results in a 3-phase approach for PTP:

* Phase 1: NTS-KE Phase (Section 3.1)

* Phase 2: PTP Unicast Transmission Negotiation Phase (Section 3.2)

* Phase 3: PTP Unicast Packet Transmission Phase (Section 3.3)

In a typical use-case, phase 1 is required to be performed at startup. In phase 2 the PTP client and the PTP server will negotiate the transmission of PTP messages which will then be delivered by the PTP server in phase 3. Whenever phase 3 ends, the PTP client must re-run phase 2 to re-request more packets. Typically, PTP clients will re-run phase 2 before the active contract ends, i.e. they request a new transmission contract from the PTP server with a new duration before the active contract expires in order to secure a continuous flow of messages from the PTP server.

```
            NTS4UPTP Protocol Overview
                                              +--------------+
                                              |              |
                                          +-> | PTP server 1 |
                                          |   |              |
                   Shared cookie          |   +--------------+
+---------------+  encryption parameters  |   +--------------+
|               |                         |   |              |
| NTS-KE Server | <-----------------------+-> | PTP server 2 |
|               |                         |   |              |
+---------------+                         |   +--------------+
     ^                                    |          .
     |                                    |          .
     | 1. Negotiate parameters,           |          .
     |    receive initial cookie,         |   +--------------+
     |    generate AEAD keys,             |   |              |
     |    and receive PTP server IP       +-> | PTP server N |
     |    addresses using "NTS Key            |              |
     |    Establishment" protocol.            +--------------+
     |                                              ^  ^
     |                                              |  |
     |                                              |  |
     |               2. Perform authenticated       |  |
     |                  unicast message negotiation |  |
     |                  using the MAC function of   |  |
     |                  the AEAD to calculate the   |  |
     |                  ICV in the                  |  |
     |                  AUTHENTICATION_TLV using    |  |
     |                  the S2C/C2S keys            |  |
     |                                              |  |
     |             +----------+                     |  |
     |             |  PTP     |                     |  |
     +-----------> |  Client  | <-------------------+  |
                   |          | <----------------------+
                   +----------+
                         3. PTP server sends unicast messages
                            as negotiated with PTP client using
                            the MAC function of the AEAD to
                            calculate the ICV in the
                            AUTHENTICATION_TLV using the S2C key;
                            Client sends DELAY_REQ messages using the
                            C2S key for the ICV
```

Figure 1: Overview of High-Level Interactions in NTS4UPTP

Phase 1 only needs to be re-run to avoid that the key lifetime expires, the PTP server stops responding or does not accept the cookie presented by the PTP client for any reasons.

3.1. Phase 1: NTS-KE Phase

In the NTS-KE Phase (Phase 1), the PTP client connects to the NTS-KE server via a secure TLS channel. Two keys are generated based on the TLS data exchange, referred to as Server-to-Client key (S2C key) and Client-to-Server key (C2S key). The NTS-KE server creates a cookie for the PTP client, which contains those two keys, the AEAD algorithm used and a Nonce. The cookie is secured by encrypting this information with a master key K. The master key is generated by a PTP server and sent to the KE server via a proprietary mechanism. The client therefore cannot decrypt a cookie as it does not know the master key. After receiving the cookies and establishing the C2S and S2C keys, the TLS connection is closed. This is identical to the NTS-KE phase as defined in [RFC8915] with the exception that the NTS-KE record type "next-protocol" points to PTP instead of NTP and two new NTS record types are introduced:

* "Cookie for Unicast PTP" provides the client with the cookie required to establish a valid NTS connection with the PTP server. This NTS record type is defined in Section 4.1.

* "PTP server Negotiation" tells the client which PTP server it MUST use. This NTS record type is defined in Section 4.2.

The MAC function of the AEAD algorithm will be used to create/ calculate the ICV in the AUTHENTICATION_TLV as defined in subclause 16.14 of [IEEE1588] (Integrated PTP Security).

For providing the cookie, an NTS-KE server MUST use a new NTS record type (New Cookie for Unicast PTP), which is identical to the NTS record type 5 (New Cookie for NTPv4) as described in [RFC8915].

The S2C key will be used in Phase 2 and 3 to calculate the ICV of the AUTHENTICATION_TLV for all PTP messages sent from the PTP server to the PTP client. The C2S key will be used in Phase 2 and 3 to calculate the ICV of the AUTHENTICATION_TLV for all PTP messages sent from the PTP client to the PTP server.

After the PTP client successfully completed Phase 1, it can enter Phase 2 by initiating the PTP unicast negotiation with the provided PTP server.

3.2. Phase 2: PTP Unicast Transmission Negotiation Phase

A unicast PTP client needs to establish a contract with a PTP server if it wants to receive SYNC and ANNOUNCE messages and to perform delay measurements by sending DELAY_REQ messages to the PTP server, which responds with DELAY_RESP messages.

The mechanism to establish these contracts between PTP client and PTP server is described in subclause 16.1 of [IEEE1588] ("Unicast message negotiation"). The basic concept requires the PTP client to send a request for each specific message type to transmit that message at a specific rate for a specific duration. The PTP server either grants the request or rejects it (for example due to capacity constraints). Each message type requires its own contract between a PTP client and a PTP server and there can only be one active contract per message type between the two nodes.

According to [IEEE1588] PTP messages can be extended by adding one or more TLVs on the end of the PTP message, and [IEEE1588] defines a number of TLVs. Here TLV stands for type-length-value. A standards development organization can also define TLVs to support specifications for extending PTP. In this case the TLV will be "ORGANIZATION_EXTENSION_PROPAGATE" or "ORGANIZATION_EXTENSION_DO_NOT_PROPAGATE" depending on whether a PTP Boundary Clock shall pass the TLV on or not. This kind of TLV MUST include a field for with the organizationID and a field with the organizationSubType. The latter MUST be unique among PTP TLVs defined by that organization.

The nature of unicast PTP requires that a PTP server maintains a list of PTP clients with active contracts. For NTS4UPTP, a server needs to store additional data. The storage entity required for storing the additional data is referred to as the PTPCLTABLE (Section 5) in this document.

In order to secure the PTP unicast transmission negotiation, PTP clients and servers use cookies and nonces, and protect the integrity of the PTP signaling messages with the integrated security mechanism based on the AUTHENTICATION_TLV described in [IEEE1588]. A PTP client initially requires a cookie for the first message it sends during this phase and will receive a nonce each time the PTP server sends a response. The initial cookie for the first request of the client is provided by the NTS-KE server in the NTS-KE Phase. For each consecutive message the PTP client sends, it MUST use the nonce received in the previous message from the PTP server and send that one back in the NONCE field of the NTS_TLV.

Due to the fact, that the S2C/C2S key pair expires, the client is forced to get new keys from the NTS-KE server. The client MAY do this at any time and MUST do it when receiving a NTS_INVALIDKEYS response from the PTP server.

Nonces and cookies are not required in the third phase, in which the transmitted PTP messages are only secured with the AUTHENTICATION_TLV. Packet rates in this phase can be very high, PTP for example allows for up to 128 SYNC packets per second sent by the

PTP server to a client. Due to the fact that PTP requires a client to successfully complete the negotiation phase, it is sufficient to protect the integrity of the messages in the transmission phase with the AUTHENTICATION_TLV.

The unicast negotiation mechanism as specified in [IEEE1588] is carried out according to the standard, but with the following addition:

* The PTP client and the PTP server MUST add an NTS TLV and an AUTHENTICATION_TLV to all signaling messages
* The NTS_TLV (Section 6) in a message sent from the PTP client to the PTP server either carries the cookie that the client obtained during the last successfully completed NTS-KE Phase, or the last nonce provided by the PTP server in its response. Additionally the ntsMsgId MUST be set to NTS_INIT for the first message sent to the PTP server after completing the NTS-KE phase and to NTS_REQUEST for all further messages.
* The AUTHENTICATION_TLV secures the REQUEST_UNICAST_TRANSMISSION_TLV and the NTS_TLV with an ICV which is calculated based on the MAC function of the AEAD algorithm and the C2S key that has been obtained during the NTS-KE phase
* All signaling messages from the PTP server to the PTP client MUST carry a new nonce and the ICV in the AUTHENTICATION_TLV is calculated based on the S2C key that the PTP server read from the decrypted cookie in the last NTS_INIT message from the client.

All PTP messages in the packet negotiation phase that do not carry an AUTHENTICATION_TLV or in which the ICV is not correct MUST be ignored by the recipient.

A PTPCLTABLE entry SHALL be stored at least for as long as the S2C/C2S key pair is valid, i.e. until KEY_PAIR_EXP has been reached. The maximum lifetime of a key pair is defined in the configuration of the PTP server and the expiration date/time is calculated when the entry in this PTP client state storage is created (Expiration date/time=creation time of PTPCLTABLE record plus configured maximum lifetime). The PTP server SHOULD erase entries from this table after the expiration time of the key pair has been reached.

The PTP server, after receiving a signaling message, will perform the following steps:

* if the request contains an NTS_TLV with ntsMsgId == NTS_INIT, it decrypts the cookie with its master key K to obtain the two C2S

and S2C keys, for all other ntsMsgId values it MUST perform a lookup into the PTPCLTABLE

* if the request contains an NTS_TLV with ntsMsgId == NTS_REQUEST, the nonce used in this signaling message is identical to the NEXT_NONCE stored for this client in the PTPCLTABLE

* if the request contains an NTS_TLV with ntsMsgId == NTS_CHALLENGE_REQUEST, CHALLENGE_EXP has not been reached and the cookie used in this signaling message is identical to the CHALLENGE_NONCE stored for this client

* If either the cookie cannot be decrypted, no matching entry could be found in the PTPCLTABLE, the nonce does not match the NEXT_NONCE or the CHALLENGE_NONCE, the message MUST be ignored. In all other cases, the following additional checks MUST be performed:

  - the ICV in the AUTHENTICATION_TLV is correct (using the C2S key from the provided cookie or from the matching entry in the PTPCLTABLE

  - the key expiration date/time has not been reached

  If the CHALLENGE_NONCE check fails (only applicable when the ntsMsgId in the received message is NTS_CHALLENGE_RESPONSE), the message MUST be ignored.

  If the key expiration check fails, the request is denied and, by setting the key lifetime field of the NTS_TLV in the response to 0 and the ntsMsgId to NTS_INVALIDKEYS, the client is told to obtain new keys and a new cookie from a NTS-KE server before it can establish a new contract with this PTP server.

If no matching entry exists in the PTPCLTABLE, or if the checks above result require an NTS_CHALLENGE_REQUEST response, any active contract (if there is one) MUST NOT be changed, canceled or otherwise modified. This is to avoid that an attacker sends an invalid request which stops the currently active contract and therefore successfully carries out a denial-of-service attack.

When a PTP server receives a NTS_INIT message with a valid cookie, the following challenge/response procedure MUST be followed:

* The PTP server will deny the REQUEST_UNICAST_TRANSMISSION_TLV request and responds with an NTS_MessageId NTS_CHALLENGE_REQUEST and a nonce which it stores as the CHALLENGE_NONCE in the PTPCLTABLE. The server will put the sequenceId from the PTP packet header of the request into the reqSeqId field of the NTS_TLV

* The client, after receiving the response with the NTS_CHALLENGE_REQUEST message ID SHALL check that the reqSeqId is identical to the sequenceId of the request it sent to the server. If this fails, the message MUST be ignored by the client.

* If the reqSeqId check is successful, the client resends the REQUEST_UNICAST_TRANSMISSION_TLV using a ntsMsgId of NTS_CHALLENGE_RESPONSE and MUST use the cookie it received with the NTS_CHALLENGE_REQUEST response.

* The server then checks that the cookie presented by the client in the NTS_CHALLENGE_RESPONSE response is identical to the CHALLENGE_NONCE. If that is true, the client can be trusted and the REQUEST_UNICAST_TRANSMISSION_TLV included in this response is considered to be a valid and trusted request.

After the successful verification of the request, the REQUEST_UNICAST_TRANSMISSION_TLV is processed according to [IEEE1588].

In case the PTP server grants the request to the client, the process moves on to the 3rd phase, the packet transmission phase. If the PTP server denies the request for whatever reason (i.e. it has no capacity or its state does not allow to reliably transmit the packets at the requested rate), it sends a unicast grant denial message, i.e. a PTP signaling message carrying a GRANT_UNICAST_TRANSMISSION TLV with the grant duration set to zero.

If a PTP client receives a GRANT_UNICAST_TRANSMISSION_TLV containing an NTS_TLV with a correct ICV and a key lifetime set to 0, it MUST delete all cookies it holds for that PTP server as well as the S2C/C2S key pair and cease all communication until it re-ran phase 1 and obtained a new key pair and a new cookie.

Using the "maximum key lifetime" configuration parameter, a PTP server operator can prioritize memory requirements and required network traffic volume. A small maximum lifetime value results in PTPCLTABLE entries being deleted earlier, requiring more NTS_CHALLENGE_REQUEST exchanges between PTP client and PTP server. A large value may result in requiring fewer such packet exchanges but increases the memory consumption because PTPCLTABLE entries have to be stored for a longer period of time.

3.3. Phase 3: PTP Unicast Packet Transmission Phase

When the transmission request has been granted by the PTP server for a specific messagetype/duration, for all messagetypes except delay responses the GM immediately starts transmitting the messages in the requested rate. DELAY_RESP messages will be sent after the client sent a DELAY_REQ message.

All unicast PTP messages sent by the PTP server to the PTP client due to an active contract SHALL be secured by an AUTHENTICATION_TLV that carries an ICV as described in [IEEE1588], subclause 16.14 (PTP Integrated Security). The ICV is created using the MAC function of the AEAD algorithm and the S2C key established between the PTP client and the NTS-KE server during the NTS-KE phase. All messages sent by the PTP client to the PTP server will be secured with the same mechanism, but using the C2S key.

All PTP messages in the packet transmission phase that do not carry an AUTHENTICATION_TLV or in which the ICV is not correct MUST be ignored by the recipient.

In order to establish a protection against replay attacks in this phase, both the PTP client and the PTP server MUST check that the sequenceId of an incoming message is larger than the sequenceId of the same PTP message type received in the previous message. If an incoming message does not pass the sequenceId check, it MUST be ignored. Both PTP client and PTP server SHOULD allow to configure the maximum difference between the sequenceId values of two consecutive messages of the same message type. They MUST gracefully handle the rollover of a sequenceId, which is a unsigned int16 value (0-65535).

To avoid that an attacker resends a PTP message with a sequenceId that has been obtained before the last rollover, additional integrity checks SHOULD be applied. The maximum packet rate is 128 packets/second. Therefore, for each PTP message type sent at the maximum rate, the sequenceId rollover happens every 512 seconds (8 minutes, 32 seconds) as a minimum. For SYNC, DELAY_REQ and ANNOUNCE messages the recipient SHOULD check that the originTimestamp in the packet does not differ more than 8 minutes from the originTimestamp of the previously received message. For DELAY_RESP messages, the receiveTimestamp SHOULD be used instead.

The packet transmission phase ends either when the contract expired or when either the PTP server or the PTP client cancels the contract.

4. New NTS record types

4.1. Cookie for Unicast PTP

The content of this NTS record is identical to record type 5 as defined in 4.1.6 of [RFC8915]. However, a NTS-KE server MUST send exactly one record of this type when PTP is negotiated as a next protocol.

4.2. Unicast PTP Server Negotiation

The PTP server Negotiation record has a Record Type number of 8. Its body consists of an ASCII-encoded [RFC0020] string. The contents of the string SHALL be either an IPv4 address, an IPv6 address, or a fully qualified domain name (FQDN). IPv4 addresses MUST be in dotted decimal notation. IPv6 addresses MUST conform to the "Text Representation of Addresses" as specified in RFC 4291 [RFC4291] and MUST NOT include [RFC6874]. If a label contains at least one non-ASCII character, it is an internationalized domain name, and an A-LABEL MUST be used as defined in Section 2.3.2.1 of RFC 5890 [RFC5890]. If the record contains a domain name, the recipient MUST treat it as a FQDN, e.g., by making sure it ends with a dot.

When PTP is negotiated as a Next Protocol and this record is sent by the server, the body specifies the hostname or IP address of the PTP unicast server with which the client MUST associate and that will accept the supplied cookies. If no record of this type is sent, the client SHALL interpret this as a directive to associate with a PTP server at the same IP address as the NTS-KE server. Servers MUST NOT send more than one record of this type. If the record contains a FQDN which resolves to multiple addresses, the client MUST choose at least one of the addresses the FQDN resolves to. The client MAY choose to use more than one address to request synchronization services from multiple unicast PTP servers in parallel.

When this record is sent by the client, it indicates that the client wishes to associate with the specified PTP server. The NTS-KE server MAY incorporate this request when deciding which PTP server Negotiation records to respond with, but honoring the client's preference is OPTIONAL. The client MUST NOT send more than one record of this type.

If the client has sent a record of this type, the NTS-KE server SHOULD reply with the same record if it is valid and the server is able to supply cookies for it. If the client has not sent any record of this type, the NTS-KE server SHOULD respond with either an PTP server address in the same family as the NTS-KE session or a FQDN that can be resolved to an address in that family, if such alternatives are available.

Servers MAY set the Critical Bit on records of this type; clients SHOULD NOT.

5. The PTP client Table

The PTP server MUST store the following data for each PTP client in a NTS PTP client Table (PTPCLTABLE):

*the S2C/C2S key pair

- the time when the validity of this key pair expires (KEY_PAIR_EXP)
- the next nonce to be expected from the client (NEXT_NONCE)
- the nonce to be expected from the client in a NTS_CHALLENGE_RESPONSE message (CHALLENGE_NONCE)
- the time when the validity of the CHALLENGE_NONCE expires (CHALLENGE_EXP)

6. The NTS_TLV

The NTS_TLV contains:

1. tlvType: ORGANIZATION_EXTENSION_DO_NOT_PROPAGATE (8000 hex)
2. lengthField: number of octets in the value + 6
3. organizationID: IETF (=00005E hex)
4. organizationSubtype: TBD (needs to be assigned)
5. ntsMsgId: NTS Message Id (see below)
6. networkProtocol: Transport type (0x01=IPv4, 0x02=IPv6, 0x03=Ethernet), unsigned int
7. tSrcAddr: Transport source address of the sender, e.g. the IP address or Ethernet MAC address, 16 octets
8. keyLifetime: Key Lifetime in seconds, unsigned int32
9. reqSeqId: sequenceId of the request, unsigned int16
10. nonce/cookie: a nonce or, if ntsMsgId == NTS_INIT, an NTS Cookie

Bits								Octets	TLV Offset
0	1	2	3	4	5	6	7		
tlvType								2	0
lengthField								2	2
organizationId								3	4
organizationSubType								3	7
ntsMsgId								1	10
networkProtocol								1	11
tSrcAddr								16	12
keyLifetime								4	28
reqSeqId								2	32
nonce/cookie								n	34

Figure 2: NTS TLV Format

The networkProtocol field allows to detect which transport protocol is in use and therefore how to interprete the tSrcAddr field. For an Ethernet MAC address, the 6 first octets of the tSrcAddr field are relevant, for an IPv4 address the first 4 octets are relevant and for an IPv6 address the full 16 octets are relevant. The enumeration is corresponding to the networkProtocol field as defined in [IEEE1588].

Please note that the size of the cookie depends on the chosen AEAD, it can therefore differ and has been placed at the end of the TLV. The lengthField allows a receipient to calculate the length of the cookie.

The ntsMsgId field MUST contain one of the following values:

* 0x01 NTS_INIT (for the first unicast negotiation message sent by the PTP client to the PTP server after completing a NTS-KE Phase in which the PTP client obtained a new key pair)

* 0x02 NTS_REQUEST (for unicast negotiation messages sent by the PTP client to the PTP server)

* 0x03 NTS_RESPONSE (for unicast negotiation messages sent by the PTP server to the PTP client)

* 0x04 NTS_CHALLENGE_REQUEST (for messages sent by the PTP server to the PTP client in case of a challenge/response procedure)

* 0x05 NTS_CHALLENGE_RESPONSE (for messages sent by the PTP client to the PTP server as a resonse to a NTS_CHALLENGE_REQUEST)

* 0x06 NTS_INVALIDKEYS (for messages sent by the PTP server to the PTP client when the S2C/C2S key pair expired or whenever the PTP server wants to force the PTP client to re-run the NTS-KE Phase)

The reqSeqId field MUST be set to 0 for all messages except those with a NTS_CHALLENGE_REQUEST message Id. In that specific case the field MUST contain the sequenceId of the message to which this NTS_CHALLENGE_REQUEST is a response.

When sending a REQUEST_UNICAST_TRANSMISSION_TLV, the PTP client will add the NTS_TLV containing a cookie for this PTP server. The key lifetime SHALL always be set to 0 for all communication from the PTP client to the PTP server.

When sending a GRANT_UNICAST_TRANSMISSION_TLV, the PTP server will add the NTS_TLV as well. If the request of the client is granted (duration > 0), the NTS_TLV will contain a new cookie and the key lifetime field SHALL represent the number of seconds the C2S and S2C keys continue to be valid.

The PTP server SHALL set a key lifetime of 0 when the lifetime of the S2C/C2S key pair expired. This allows the PTP server to force the client to re-start and obtain fresh keys and cookies from the NTS-KE server. When sending an NTS_TLV with the key lifetime set to 0, the PTP server SHALL use the S2C key of the expired key pair to form the ICV in the AUTHENTICATION_TLV, allowing the client to verify that the message has been sent by the PTP server.

When a client receives any message with a valid ICV but a key lifetime of 0 in the NTS_TLV, it SHALL delete all cookies cease to send any messages to the PTP server and only restores communication with it after it obtained a new S2C/C2S key pair and a new set of cookies from the NTS-KE server.

7. IANA Considerations

RFC EDITOR: A new entry for Unicast PTP is required in the IANA Network Time Security Next Protocols Registry. The authors propose to add an entry with the Protocol ID = 1, the Protocol Name = "Unicast Precision Time Protocol" and a Reference to this draft.

Please remove this comment before publishing and replace it with the data for the newly created entry from IANA.

8. Security Considerations

8.1. Threat Model

The procedures and mechanisms described in this draft protect against the following scenarios:

* Man-in-the-Middle (MITM) attack: All PTP nodes can verify that PTP messages received from another PTP server have not been modified in transit by an attacker. This is achieved by verifying that the ICV in the AUTHENTICATION_TLV of every PTP message received is valid and has been created using the MAC function of the AEAD algorithm and the C2S or S2C key as established during the NTS-KE phase.

* Phase 2 Replay Attack (resending unmodified PTP unicast negotiation messages): PTP message integrity can be secured with the AUTHENTICATION_TLV. However, with PTP this mechanism does not protect the transport protocol header and could therefore be used by an attacker to intercept a PTP message and then resending it using the same or a different source address. Or it could be resend at a later time to repeat a request or cancel an active contract. Resending it with the same address can be used to try and establish a contract for a unicast PTP client that is no longer requiring the service, requires the service in a different form (e.g. different message rates) or it can be used to cancel a contract (when resending a CANCEL_UNICAST_TRANSMISSION_TLV after the client established a new contract). This can interrupt a required service for a PTP client or result in the PTP server unnecessarily consuming resources. By storing the nonce that has been provided by the server and that MUST be used by the client in its next request (NEXT_NONCE in PTPCLTABLE), an unmodified resent REQUEST_UNICAST_TRANSMISSION_TLV will not be granted, despite the fact that the ICV in the AUTHENTICATION_TLV is valid. T o provide a robust defense against replaying NTS_INIT messages, the challenge/response mechanism (NTS_CHALLENGE_REQUEST/ NTS_CHALLENGE_RESPONSE) will require a PTP client to apply the correct C2S key and therefore protects against replaying a previously sent valid message with a cookie that has been encrypted with a still valid master key K.

* Phase 3 Replay Attack (resending unmodified PTP unicast messages): In Phase 3 the PTP server is sending PTP SYNC and ANNOUNCE messages to the PTP client and the PTP client is sending DELAY_REQ messages to the PTP server, which responds with DELAY_RESP messages. An attacker could resend any of these

packets. For SYNC messages, that would result in the PTP client receiving "old time", i.e. the timestamps in such a message would be outdated and could disrupt time synchronization of the PTP client. A resent ANNOUNCE message could carry outdated information and therefore could force the PTP client to drop this server as a valid time source or distrupt the protocol in other ways. Resending DELAY_REQ messages could consume resources on the PTP server and, if the server would send DELAY_RESP message, on the PTP client as well. The client could also be negatively affected by resent DELAY_RESP messages that carry outdated time. A valid protection against such an attack is checking the sequenceId of each incoming message and by applying additional integrity checks to messages passing the sequenceId checks. See Section 3.3 for a detailed description of how this can be achieved.

* Amplification Attack/Distributed Denial of Service: Resending a REQUEST_UNICAST_TRANSMISSION_TLV with a different source IP address could result in the PTP server sending PTP messages to IP adresses that do not expect and require receiving these messages. Due to the nature of unicast PTP, the traffic amplification porential is very high because one PTP signaling message containing a REQUEST_UNICAST_TRANSMISSION_TLV can generate thousands of PTP messages from the PTP server to the source IP address used in the request message. This is mitigated by including the IP address of the originator of a message as a field (tSrcAddr) in the NTS_TLV. The TLV as part of the PTP message is protected by the ICV in the AUTHENTICATION_TLV and therefore a modified source address can be detected.

8.2. General Security Features

In addtion to the above outlined protection mechanisms against specific attack scenarios, this draft also includes a generic security feature:

* Key Freshness: The expiration of C2S/S2C key pairs requires clients to obtain a new key pair in a configurable interval, which limits the time an attacker has to break those keys.

9. Delay Attacks

If an attacker gains control over a part of the network infrastructure on the path between clients and server, it could be possible to delay the forwarding of unicast PTP messages without modifying them. Applying such a delay only in one direction (e.g. for SYNC packets sent from the server to the client) would create an asymmetry in the delay calculation and as a result the error in the delay calculation would cause a timing error on the client. This kind of attack cannot be mitigated by NTS4UPTP as the cryptographic protection only allows to ensure the integrity of messages, which is not corrupted by a delay attack.

In addition to securing the network infrastructure (i.e. routers and switches) against this threat, another possible mitigation strategy for the client is to check the calculated delay against a static limit (which could be configurable by the user or is defined by the requirements of the application) or a dynamic limit, which the client could determine periodically for example by applying suitable statistical methods to determine a change in the calculated delay that indicates that the potential time error would exceed the sync requirements of the application.

10. Acknowledgements

The authors would like to thank the following contributors for their valuable feedback:

*Martin Langer, MSc and Prof. Dr.-Ing. Rainer Bermbach, Ostfalia University

11. References

11.1. Normative References

[RFC0020] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <https://www.rfc-editor.org/info/rfc20>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[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, <https://www.rfc-editor.org/info/rfc5905>.

[RFC8915] Franke, D., Sibold, D., Teichel, K., Dansarie, M., and R. Sundblad, "Network Time Security for the Network Time

Protocol", RFC 8915, DOI 10.17487/RFC8915, September 2020, <https://www.rfc-editor.org/info/rfc8915>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/info/rfc5890>.

[RFC6874] Carpenter, B., Cheshire, S., and R. Hinden, "Representing IPv6 Zone Identifiers in Address Literals and Uniform Resource Identifiers", RFC 6874, DOI 10.17487/RFC6874, February 2013, <https://www.rfc-editor.org/info/rfc6874>.

11.2. Informative References

[IEEE1588] IEEE, "IEEE Std 1588-2019 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control", IEEE Standard 1588, 2019, <https://standards.ieee.org/content/ieee-standards/en/standard/1588-2019.html>.

Authors' Addresses

Heiko Gerstung
Meinberg Funkuhren GmbH&Co.KG
Lange Wand 9
31812 Bad Pyrmont
Germany

Phone: +49 5281 9309 0
Email: heiko.gerstung@meinberg.de
URI: http://www.meinbergglobal.com/

Marius Rohde
Meinberg Funkuhren GmbH&Co.KG
Lange Wand 9
31812 Bad Pyrmont
Germany

Phone: +49 5281 9309 0
Email: marius.rohde@meinberg.de
URI: http://www.meinbergglobal.com/

Douglas Arnold
Meinberg USA Inc.
100 Stony Point Road Suite 110
Santa Rosa, CA 95401
United States of America

Phone: +1 877-PTP-1588

Email: doug.arnold@meinberg-usa.com
URI: http://www.meinbergglobal.com/