NTP Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 20, 2014

D. Sibold
PTB
S. Roettger

K. Teichel
PTB
March 19, 2014

Network Time Security
draft-ietf-ntp-network-time-security-03.txt

Abstract

This document describes the Network Time Security (NTS) protocol that enables secure authentication of time servers using Network Time Protocol (NTP) or Precision Time Protocol (PTP). Its design considers the special requirements of precise timekeeping, which are described in Security Requirements of Time Protocols in Packet Switched Networks [I-D.ietf-tictoc-security-requirements].

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 September 20, 2014.

Table of Contents

1. Introduction

Time synchronization protocols are utilized more and more to synchronize clocks in networked infrastructures. The reliable performance of such infrastructures can be degraded seriously by successful attacks against the time synchronization protocol. Therefore, time synchronization protocols applied in critical infrastructures have to provide security measures to defeat possible adversaries. Consequently, the widespread Network Time Protocol (NTP) [RFC5905] was supplemented by the autokey protocol [RFC5906] which shall ensure authenticity of the NTP server and integrity of the protocol packets. Unfortunately, the autokey protocol exhibits various severe security vulnerabilities as revealed in a thorough analysis of the protocol [Roettger]. For the Precision Time Protocol (PTP), Annex K of the standard document IEEE 1588 [IEEE1588] defines an informative security protocol that is still in experimental state.

Because of autokey's security vulnerabilities and the absence of a standardized security protocol for PTP, these protocols cannot be applied in environments in which compliance requirements demand authenticity and integrity protection. This document specifies a security protocol which ensures authenticity of the time server via a Public Key Infrastructure (PKI) and integrity of the time

synchronization protocol packets and which therefore enables the usage of NTP and PTP in such environments.

The protocol is specified with the prerequisite in mind that precise timekeeping can only be accomplished with stateless time synchronization communication, which excludes the utilization of standard security protocols like IPsec or TLS for time synchronization messages. This prerequisite corresponds with the requirement that a security mechanism for timekeeping must be designed in such a way that it does not degrade the quality of the time transfer [I-D.ietf-tictoc-security-requirements].

Note:

The intent is to formulate the protocol to be applicable to NTP as well as PTP. In the current state the draft focuses on the application to NTP.

2. Security Threats

A profound analysis of security threats and requirements for NTP and PTP can be found in the I-D [I-D.ietf-tictoc-security-requirements].

3. Objectives

The objectives of the NTS specifications are as follows:

- Authenticity: NTS enables the client to authenticate its time server.
- Integrity: NTS protects the integrity of time synchronization protocol packets via a message authentication code (MAC).
- Confidentiality: NTS does not provide confidentiality protection of the time synchronization packets.
- Modes of operation: All operational modes of NTP are supported.
- Operational modes of PTP should be supported as far as possible.
- Hybrid mode: Both secure and insecure communication modes are possible for NTP servers and clients, respectively.
- Compatibility:
  - Unsecured NTP associations shall not be affected.

* An NTP server that does not support NTS shall not be affected by NTS authentication requests.

4. Terms and Abbreviations

o TESLA: Timed Efficient Stream Loss-Tolerant Authentication

5. NTS Overview

5.1. Symmetric and Client/Server Mode

Authenticity of the time server is verified once by utilization of X.509 certificates. Authenticity and integrity of the NTP packets are then ensured by a Message Authentication Code (MAC), which is attached to the NTP packet. The calculation of the MAC includes the whole NTP packet and the cookie which is shared between client and server. It is calculated according to:

cookie = MSB_128 (HMAC(server seed, H(public key of client))),

with the server seed as key, where H is a hash function, and where the function MSB_128 cuts off the 128 most significant bits of the result of the HMAC function. The server seed is a 128 bit random value of the server, which has to be kept secret. The cookie never changes as long as the server seed stays the same, but the server seed has to be refreshed periodically in order to provide key freshness as required in [I-D.ietf-tictoc-security-requirements]. See Section 9 for details on the seed refresh and Section 7.1.1 for the client's reaction to it.

The server does not keep a state of the client. Therefore it has to recalculate the cookie each time it receives a request from the client. To this end, the client has to attach the hash value of its public key to each request (see Section 6.4).

5.2. Broadcast Mode

Just as in the case of the client server mode and symmetric mode, authenticity and integrity of the NTP packets are ensured by a MAC, which is attached to the NTP packet by the sender. The verification of the authenticity is based on the TESLA protocol, in particular on its "Not Re-using Keys" scheme, see section 3.7.2 of [RFC4082]. TESLA is based on a one-way chain of keys, where each key is the output of a one-way function applied on the previous key in the chain. The last element of the chain is shared securely with all clients. The server splits time into intervals of uniform duration and assigns each key to an interval in reverse order, starting with the penultimate. At each time interval, the server sends an NTP

broadcast packet appended by a MAC, calculated using the corresponding key, and the key of the previous disclosure interval. The client verifies the MAC by buffering the packet until the disclosure of the key in its associated disclosure interval. In order to be able to verify the validity of the key, the client has to be loosely time synchronized to the server. This has to be accomplished during the initial client server exchange between broadcast client and server. For a more detailed description of the TESLA protocol see Appendix D.

6. Protocol Messages

Note that this section currently describes realization of the message format of NTS only for its utilization for NTP, in which the NTS specific data are enclosed in extension fields on top of NTP packets. A specification of NTS messages for PTP would have to be developed accordingly.

The steps described in Section 6.1 - Section 6.4 belong to the unicast mode, while Section 6.5 and Section 6.6 explain the steps involved in the broadcast mode of NTS.

6.1. Association Messages

In this step, the hash and signature algorithms that are used for the rest of the protocol are negotiated.

6.1.1. Message type: "client_assoc"

The protocol sequence starts with the client sending an association message, called client_assoc. This message contains

- the version number of NTS that the client wants to use (this SHOULD be the highest version number that it supports),
- the hostname of the client,
- a selection of hash algorithms, and
- a selection of accepted algorithms for the signatures.

For NTP, this message is realized as a packet with an extension field of type "association", which contains all this data.

6.1.2. Message type: "server_assoc"

This message is sent by the server upon receipt of client_assoc. It contains

- the version number used for the rest of the protocol (which SHOULD be determined as the minimum over the client's suggestion in the client_assoc and the highest supported by the server),
- the hostname of the server, and
- the server's choice of algorithm for the signatures and cryptographic hash algorithm, both of which MUST be chosen from the client's proposals.

In the case of NTP, the data is enclosed in a packet's extension field, also of type "association".

6.2. Certificate Messages

In this step, the client receives the certification chain up to a trusted anchor. With the established certification chain the client is able to verify the server signatures and, hence, the authenticity of the server messages with extension fields is ensured.

Discussion:

> Note that in this step the client validates the authenticity of its immediate NTP server only. It does not recursively validate the authenticity of each NTP server on the time synchronization chain. Recursive authentication (and authorization) as formulated in [I-D.ietf-tictoc-security-requirements] depends on the chosen trust anchor.

6.2.1. Message type: "client_cert"

This message is sent by the client, after it successfully verified the content of the received server_assoc message (see Section 7.1.1). It contains

- the negotiated version number,
- the client's hostname, and
- the signature algorithm negotiated during the association messages.

It is realized as an NTP packet with extension field of type "certificate request" for the necessary data.

6.2.2. Message type: "server_cert"

This message is sent by the server, upon receipt of a client_cert message, if the version number and choice of methods communicated in that message are actually supported by the server. It contains

- all the information necessary to authenticate the server to the client. This is a chain of certificates, which starts at the server and goes up to a trusted authority, where each certificate MUST be certified by the one directly following it.

This message is realized for NTP as a packet with extension field of type "certificate" which holds the certification data.

6.3. Cookie Messages

During this step, the server transmits a secret cookie to the client securely. The cookie will be used for integrity protection during unicast time synchronization.

6.3.1. Message type: "client_cook"

This message is sent by the client, upon successful authentication of the server. In this message, the client requests a cookie from the server. It contains

- the negotiated version number,
- the hash algorithm H negotiated between client and server during the association messages,
- the client's public key.

For NTP, an extension field of type "cookie request" holds the listed data.

6.3.2. Message type: "server_cook"

This message is sent by the server, upon receipt of a client_cook message. The hash of the client's public key, as included in client_cook, is used by the server to calculate the cookie (see Section 5.1). This message contains

- a concatenated pair, encrypted with the client's public key, where the pair consists of

* the cookie, and

* a signature of the cookie signed with the server's private key.

In the case of NTP, this is a packet with an extension field of type "cookie transmit".

6.4. Unicast Time Synchronisation Messages

In this step, the usual time synchronization process is executed, with the addition of integrity protection for all messages that the server sends. This step can be repeated as often as the client desires and as long as the integrity of the server's time responses is verified successfully. Secure time synchronization by repetition of this step is the goal of a unicast run.

6.4.1. Message type: "time_request"

This message is sent by the client when it requests time exchange. To send this message, the client MUST have received server_cook and successfully verified the cookie via the server's signature. It contains

o the negotiated version number,

o the time synchronization data that the client wants to transmit,

o a 128-bit nonce,

o the negotiated hash algorithm H,

o the hash of the client's public key under H.

It is realized as an NTP packet with the time synchronization data and an additional extension field of type "time request" for the rest of the information.

6.4.2. Message type: "time_response"

This message is sent by the server, after it received a time_request message. The server uses the hash of the client's public key and the transmitted hash algorithm to recalculate the cookie for the client. This message contains

o the server's time synchronization response data,

o the nonce transmitted in time_request,

- a MAC (generated with the cookie as key) for verification of the above.

It is realized as an NTP packet with the necessary time synchronization data and with a new extension field of type "time response". This packet has an appended MAC that is generated over the time synchronization data and the extension field, with the cookie as the key.

6.5. Broadcast Parameter Messages

In this step, the client receives the necessary information to execute the TESLA protocol in a secured broadcast association. The client can only initiate a secure broadcast association after a successful unicast run, see Section 7.1.2.

See Appendix D for more details on TESLA.

6.5.1. Message type: "client_bpar"

This message is sent by the client in order to establish a secured time broadcast association with the server. It contains

- the version number negotiated during association in unicast mode,
- the client's hostname, and
- the signature algorithm negotiated during unicast.

For NTP, this message is realized as a packet with an extension field of type "broadcast request".

6.5.2. Message type: "server_bpar"

This message is sent by the server upon receipt of a client_bpar message during the broadcast loop of the server. It contains

- the one-way function used for building the one-way key chain,
- the last key of the one-way key chain, and
- the disclosure schedule of the keys. This contains:
  - time interval duration,
  - the disclosure delay (number of intervals between use and disclosure of a key),

* the time at which the next time interval will start, and

* the next interval's associated index.

- The message also contains a signature signed by the server with its private key, verifying all the data listed above.

It is realized for NTP as a packet with an extension field of type "broadcast parameters", which contains all the given data.

6.6. Broadcast Message

In this step, the server keeps sending broadcast time synchronization messages to all participating clients.

6.6.1. Message type: "server_broad"

This message is sent by the server over the course of its broadcast schedule. It is part of any broadcast association. It contains

- time broadcast data,

- the index that belongs to the current interval (and therefore identifies the current, yet undisclosed key)

- the disclosed key of the previous disclosure interval (current time interval minus disclosure delay).

- a MAC, calculated with the key for the current time interval, verifying the time data

The message is realized as an NTP broadcast packet with the time broadcast data and with an extension field of type "broadcast message", which contains the rest of the listed data. The NTP packet is then appended by a MAC verifying the time data, but not the extension field.

7. Protocol Sequence

7.1. The client

7.1.1. The client in unicast mode

For a unicast run, the client performs the following steps:

1. It sends a client_assoc message to the server.

2. It waits for a reply in the form of a server_assoc message. After receipt of the message it performs the following checks:

   * The message MUST contain a conform version number.

   * The client has to verify that the server has chosen the signature and hash algorithms from its proposal sent in the client_assoc message.

   If one of the checks fails, the client MUST abort the run.

3. The client then sends a client_cert message to the server.

4. It awaits a reply in the form of a server_cert message and performs an authenticity check. If this check fails, the client MUST abort the run.

5. Next, it sends a client_cook message to the server.

6. It awaits a reply in the form of a server_cook message; upon receipt it executes the following actions:

   * It decrypts the message with its own private key.

   * It checks that the decrypted message has the format of a 128 bit Cookie concatenated with its own signature value, verifiable with the server's public key.

   If the check fails, the client MAY abort the run.

7. The client sends a time_request message to the server.

8. It awaits a reply in the form of a time_response message. Upon receipt, it checks:

   * that the transmitted nonce belongs to the previous time_request message and .

   * that the appended MAC verifies the time data and the transmitted nonce.

   If the nonce is invalid, the client MUST ignore this time_response message. If the MAC is invalid, the client MUST do one of the following: abort the run or go back to step 5 (because the cookie might have changed due to a server seed refresh). If both checks are successful, the client SHOULD continue time synchronization by going back to step 7.

The client's behaviour in unicast mode is also expressed in Figure 1.

7.1.2. The client in broadcast mode

To establish a secure broadcast association with a broadcast server, the client MUST initially authenticate the broadcast server and securely synchronize its time to it up to an upper bound for its time offset in unicast mode. After that, the client performs the following steps:

1. It sends a client_bpar message to the server.

2. It waits for a reply in the form of a server_bpar message after which it performs the following checks:

   * The message must contain all the necessary information for the TESLA protocol, as listed in Section 6.5.2.

   * Verification of the message's signature.

   If any information is missing or cannot be verified as signed by the server, the client MUST abort the broadcast run.

3. The client awaits time synchronization data in the form of a server_broadcast message. Upon receipt, it performs the following checks:

   1. Proof that the MAC is based on a key that is not yet disclosed. This is achieved via a disclosure schedule, so this is where loose time synchronization is required. If verified the packet will be buffered for later authentication. Otherwise, the client MUST discard it. Note that the time information included in the packet will not be used for synchronization until its authenticity could be verified.

   2. The client checks whether it already knows the disclosed key. If so, the client SHOULD discard the packet to avoid a buffer overrun. If not, the client verifies that the disclosed key belongs to the one-way key chain by applying the one-way function until equality with a previous disclosed key is verified. If falsified, the client MUST discard the packet.

   3. If the disclosed key is legitimate the client verifies the authenticity of any packet that it received during the corresponding time interval. If authenticity of a packet is verified it is released from the buffer and the packet's time information can be utilized. If the verification fails

authenticity is no longer given. In this case the client MUST request authentic time from the server by means of a unicast time request message.

See RFC 4082[RFC4082] for a detailed description of the packet verification process.

The client's behaviour in broadcast mode can also be seen in Figure 2.

7.2. The server

The server's behaviour is not as easy to express in sequential terms as the client's, not even for a single association with one client. This is because the server does not keep state of any connection.

7.2.1. The server in unicast mode

A broadcast server MUST also support unicast mode, in order to provide the initial time synchronization is a precondition for any broadcast association. To support unicast mode, the server MUST be ready to perform the following actions:

- Upon receipt of a client_assoc message, the server constructs and sends a reply in the form of a server_assoc message as described in Section 6.1.2.
- Upon receipt of a client_cert message, the server checks whether it supports the given signature algorithm. If so, it constructs and sends a server_cert message as described in Section 6.2.2.
- Upon receipt of a client_cook message, the server calculates the cookie according to the formula given in Section 5.1. With this, it constructs a server_cook message as described in Section 6.3.2.
- Upon receipt of a time_request message, the server re-calculates the cookie, then computes the necessary time synchronization data and constructs a time_response message as given in Section 6.4.2.

Also, it must adhere to the rule of server seed refreshing, as given in [1]. More information on that can be found in Section 7.3.

7.2.2. The server in broadcast mode

To support NTS broadcast, the server MUST be ready to perform the following actions:

- Upon receipt of a client_bpar message, the server constructs and sends a server_bpar message as described in Section 6.5.2.

- The server follows the TESLA protocol in all other aspects, by regularly sending server_broad messages as described in Section 6.6.1, adhering to its own disclosure schedule.

It is also the server's responsibility to watch for the expiration date of the one-way key chain and generate a new key chain accordingly.

7.3. Server Seed Refresh

According to the requirements in [I-D.ietf-tictoc-security-requirements] the server has to refresh its server seed periodically. As a consequence the cookie used in the time request messages becomes invalid. In this case the client cannot verify the attached MAC and has to respond accordingly by re-initiating the protocol with a cookie request (Section 6.3). This is true for the unicast and broadcast mode, respectively.

Additionally, in broadcast mode the client has to restart the broadcast sequence with a time request message if the one-way key chain expires.

During certificate message exchange the client reads the expiration date of the period of validity of the server certificate. The client MAY restart the protocol sequence with the association message before the server certificate expires.

8. Hash Algorithms and MAC Generation

8.1. Hash Algorithms

Hash algorithms are used at different points: calculation of the cookie and the MAC, and hashing of the public key. Client and server negotiate a hash algorithm H during the association message exchange (Section 6.1) at the beginning of a unicast run. The selected algorithm H is used for all hashing processes in that run.

In broadcast mode, hash algorithms are used as pseudo random functions to construct the one-way key chain. Here, the utilized hash algorithm is communicated by the server and non-negotiable.

The list of the hash algorithms supported by the server has to fulfill the following requirements:

- it MUST NOT include MD5 or weaker algorithms,

- it MUST include SHA-256 or stronger algorithms.

8.2. MAC Calculation

For the calculation of the MAC client and server are using a Keyed-Hash Message Authentication Code (HMAC) approach [RFC2104]. The HMAC is generated with the hash algorithm specified by the client (see Section 8.1).

9. Server Seed Considerations

The server has to calculate a random seed which has to be kept secret and which MUST be changed periodically. The server MUST generate a seed for each supported hash algorithm.

9.1. Server Seed Algorithm

9.2. Server Seed Live Time

10. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

11. Security Considerations

11.1. Initial Verification of the Server Certificates

The client has to verify the validity of the certificates during the certification message exchange (Section 6.2). Since it generally has no reliable time during this initial communication phase, it is impossible to verify the period of validity of the certificates. Therefore, the client MUST use one of the following approaches:

- The validity of the certificates is preconditioned. Usually this will be the case in corporate networks.

- The client ensures that the certificates are not revoked. To this end, the client uses the Online Certificate Status Protocol (OCSP) defined in [RFC6277].

- The client requests a different service to get an initial time stamp in order to be able to verify the certificates' periods of validity. To this end, it can, e.g., use a secure shell connection to a reliable host. Another alternative is to request

a time stamp from a Time Stamping Authority (TSA) by means of the Time-Stamp Protocol (TSP) defined in [RFC3161].

11.2. Revocation of Server Certificates

According to Section 7.3, it is the client's responsibility to initiate a new association with the server after the server's certificate expires. To this end the client reads the expiration date of the certificate during the certificate message exchange (Section 6.2). Besides, certificates may also be revoked prior to the normal expiration date. To increase security the client MAY verify the state of the server's certificate via OCSP periodically.

11.3. Usage of NTP Pools

The certification based authentication scheme described in Section 6 is not applicable to the concept of NTP pools. Therefore, NTS is not able to provide secure usage of NTP pools.

11.4. Denial-of-Service in Broadcast Mode

TESLA authentication buffers packets for delayed authentication. This makes the protocol vulnerable to flooding attacks, causing the client to buffer excessive numbers of packets. To add stronger DoS protection to the protocol client and server SHALL use the "Not Re-using Keys" scheme of TESLA as pointed out in section 3.7.2 of RFC 4082 [RFC4082]. In this scheme the server never uses a key for the MAC generation more than once. Therefore the client can discard any packet that contains a disclosed key it knows already, thus preventing memory flooding attacks.

Note, an alternative approach to enhance TESLA's resistance against DoS attacks involves the addition of a group MAC to each packet. This requires the exchange of an additional shared key common to the whole group. This adds additional complexity to the protocol and hence is currently not considered in this document.

12. Acknowledgements

The authors would like to thank David Mills and Kurt Roeckx for discussions and comments on the design of NTS. Also, thanks to Harlan Stenn for his technical review and specific text contributions to this document.

13. References

13.1. Normative References

[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.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

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

[RFC3161] Adams, C., Cain, P., Pinkas, D., and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, August 2001.

[RFC4082] Perrig, A., Song, D., Canetti, R., Tygar, J., and B. Briscoe, "Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction", RFC 4082, June 2005.

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

[RFC5906] Haberman, B. and D. Mills, "Network Time Protocol Version 4: Autokey Specification", RFC 5906, June 2010.

[RFC6277] Santesson, S. and P. Hallam-Baker, "Online Certificate Status Protocol Algorithm Agility", RFC 6277, June 2011.

13.2. Informative References

[I-D.ietf-tictoc-security-requirements]
Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", draft-ietf-tictoc-security-requirements-05 (work in progress), April 2013.

[Roettger]
Roettger, S., "Analysis of the NTP Autokey Procedures", February 2012.

13.3. URIs

[1] I-D.ietf-tictoc-security-requirements

Appendix A. Flow Diagrams of Client Behaviour

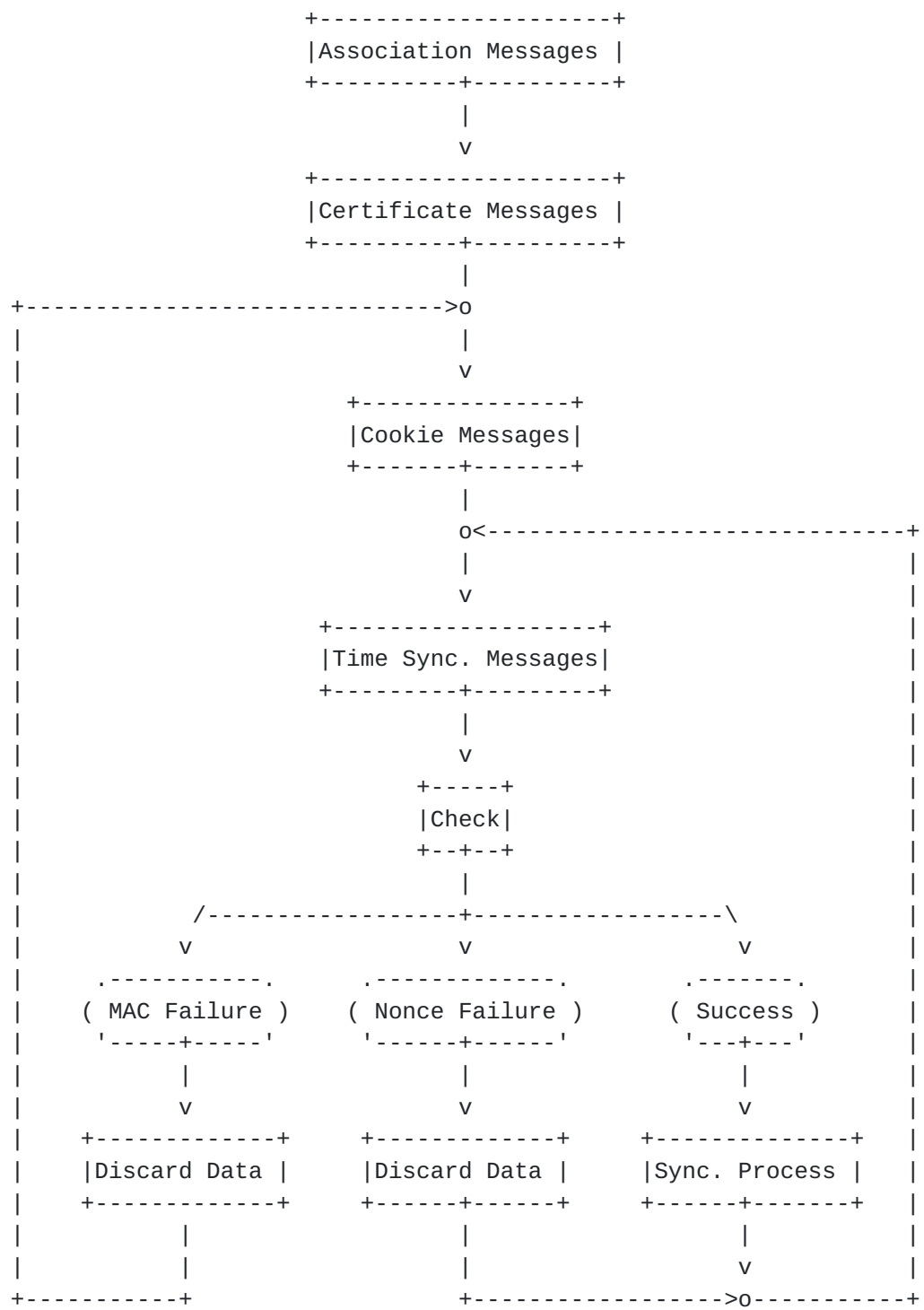

Figure 1: The client's behaviour in NTS unicast mode.

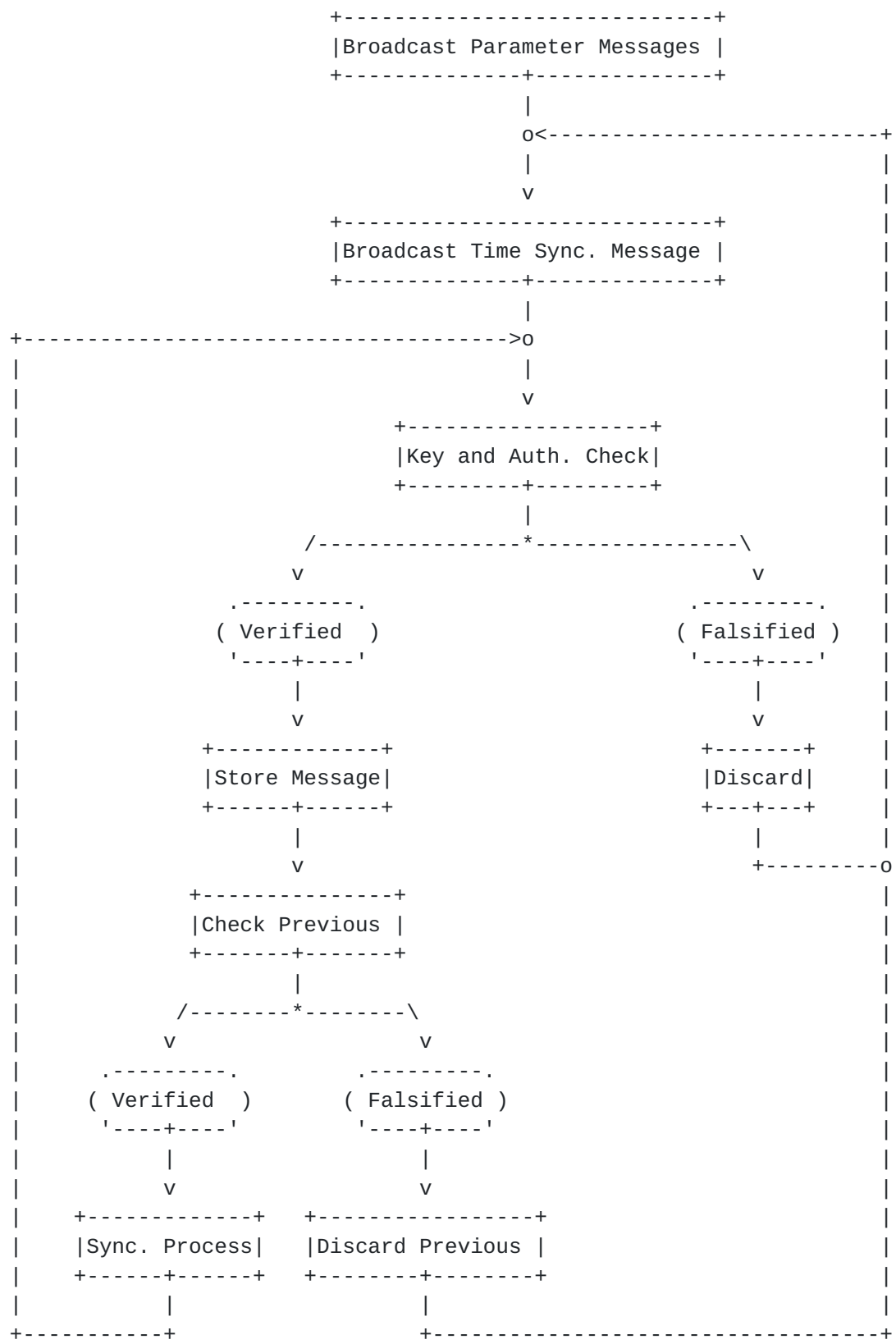

Figure 2: The client's behaviour in NTS broadcast mode.

Appendix B. Extension fields

In Section 6, some new extension fields for NTP packets are introduced. They are listed here again, for reference.

name	used in
"association"	client_assoc
	server_assoc
"certificate request"	client_cert
"certificate"	server_cert
"cookie request"	client_cook
"cookie transmit"	server_cook
"time request"	time_request
"time response"	time_response
"broadcast request"	client_bpar
"broadcast parameters"	server_bpar
"broadcast message"	server_broad

Appendix C. TICTOC Security Requirements

The following table compares the NTS specifications against the TICTOC security requirements [I-D.ietf-tictoc-security-requirements].

Section	Requirement from I-D tictoc security-requirements-05	Requirement level	NTS
5.1.1	Authentication of Servers	MUST	OK
5.1.1	Authorization of Servers	MUST	OK
5.1.2	Recursive Authentication of Servers (Stratum 1)	MUST	OK
5.1.2	Recursive Authorization of Servers (Stratum 1)	MUST	OK

5.1.3	Authentication and Authorization of Slaves	MAY	-
5.2	Integrity protection.	MUST	OK
5.3	Protection against DoS attacks	SHOULD	OK
5.4	Replay protection	MUST	OK
5.5.1	Key freshness.	MUST	OK
5.5.2	Security association.	SHOULD	OK
5.5.3	Unicast and multicast associations.	SHOULD	OK
5.6	Performance: no degradation in quality of time transfer.	MUST	OK
	Performance: lightweight computation	SHOULD	OK
	Performance: storage, bandwidth	SHOULD	OK
5.7	Confidentiality protection	MAY	NO
5.8	Protection against Packet Delay and Interception Attacks	SHOULD	NA*)
5.9.1	Secure mode	MUST	-
5.9.2	Hybrid mode	MAY	-

*) Ensured by NTP via multi-source configuration.

Comparsion of NTS sepecification against TICTOC security requirements.

Appendix D. Broadcast Mode

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

Email: stephen.roettger@googlemail.com

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

Email: kristof.teichel@ptb.de