

NTP Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 7, 2015

D. Sibold  
PTB  
S. Roettger  
Google Inc  
K. Teichel  
PTB  
March 06, 2015

## Using the Network Time Security Specification to Secure the Network Time Protocol

[draft-ietf-ntp-using-nts-for-ntp-00.txt](#)

### Abstract

This document describes how to use the measures described in the Network Time Security (NTS) specification to secure time synchronization with servers using the Network Time Protocol (NTP).

### 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 7, 2015.

### Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">3</a>
<a href="#">2.</a>	Objectives . . . . .	<a href="#">3</a>
<a href="#">3.</a>	Terms and Abbreviations . . . . .	<a href="#">4</a>
<a href="#">4.</a>	Overview of NTS-Secured NTP . . . . .	<a href="#">4</a>
<a href="#">4.1.</a>	Symmetric and Client/Server Mode . . . . .	<a href="#">4</a>
<a href="#">4.2.</a>	Broadcast Mode . . . . .	<a href="#">4</a>
<a href="#">5.</a>	Protocol Sequence . . . . .	<a href="#">4</a>
<a href="#">5.1.</a>	The Client . . . . .	<a href="#">4</a>
<a href="#">5.1.1.</a>	The Client in Unicast Mode . . . . .	<a href="#">4</a>
<a href="#">5.1.2.</a>	The Client in Broadcast Mode . . . . .	<a href="#">6</a>
<a href="#">5.2.</a>	The Server . . . . .	<a href="#">8</a>
<a href="#">5.2.1.</a>	The Server in Unicast Mode . . . . .	<a href="#">8</a>
<a href="#">5.2.2.</a>	The Server in Broadcast Mode . . . . .	<a href="#">9</a>
<a href="#">6.</a>	Implementation Notes: ASN.1 Structures and Use of the CMS . .	<a href="#">9</a>
<a href="#">6.1.</a>	Unicast Messages . . . . .	<a href="#">9</a>
<a href="#">6.1.1.</a>	Association Messages . . . . .	<a href="#">9</a>
<a href="#">6.1.2.</a>	Cookie Messages . . . . .	<a href="#">10</a>
<a href="#">6.1.3.</a>	Time Synchronization Messages . . . . .	<a href="#">10</a>
<a href="#">6.2.</a>	Broadcast Messages . . . . .	<a href="#">11</a>
<a href="#">6.2.1.</a>	Broadcast Parameter Messages . . . . .	<a href="#">11</a>
<a href="#">6.2.2.</a>	Broadcast Time Synchronization Message . . . . .	<a href="#">11</a>
<a href="#">6.2.3.</a>	Broadcast Keycheck . . . . .	<a href="#">11</a>
<a href="#">7.</a>	Security Considerations . . . . .	<a href="#">12</a>
<a href="#">7.1.</a>	Usage of NTP Pools . . . . .	<a href="#">12</a>
<a href="#">7.2.</a>	Server Seed Lifetime . . . . .	<a href="#">12</a>
<a href="#">7.3.</a>	Supported Hash Algorithms . . . . .	<a href="#">12</a>
<a href="#">8.</a>	Acknowledgements . . . . .	<a href="#">12</a>
<a href="#">9.</a>	References . . . . .	<a href="#">12</a>
<a href="#">9.1.</a>	Normative References . . . . .	<a href="#">12</a>
<a href="#">9.2.</a>	Informative References . . . . .	<a href="#">13</a>
<a href="#">Appendix A.</a>	Flow Diagrams of Client Behaviour . . . . .	<a href="#">13</a>
	Authors' Addresses . . . . .	<a href="#">16</a>



## **1. Introduction**

One of the most popular time synchronization protocols, the Network Time Protocol (NTP) [[RFC5905](#)], currently does not provide adequate intrinsic security precautions. The Network Time Security draft [[I-D.ietf-ntp-network-time-security](#)] specifies security measures which can be used to enable time synchronization protocols to verify authenticity of the time server and integrity of the time synchronization protocol packets.

This document provides detail on how to specifically use those measures to secure time synchronization between NTP clients and servers.

## **2. Objectives**

The objectives of the NTS specification are as follows:

- o Authenticity: NTS enables an NTP client to authenticate its time server(s).
- o Integrity: NTS protects the integrity of NTP time synchronization protocol packets via a message authentication code (MAC).
- o Confidentiality: NTS does not provide confidentiality protection of the time synchronization packets.
- o Authorization: NTS optionally enables the server to verify the client's authorization.
- o Request-Response-Consistency: NTS enables a client to match an incoming response to a request it has sent. NTS also enables the client to deduce from the response whether its request to the server has arrived without alteration.
- o Modes of operation: Both the unicast and the broadcast mode of NTP are supported.
- o Hybrid mode: Both secure and insecure communication modes are possible for both NTP servers and clients.
- o Compatibility:
  - \* Unsecured NTP associations are not be affected.
  - \* An NTP server that does not support NTS are not affected by NTS-secured authentication requests.



### **3. Terms and Abbreviations**

MITM    Man In The Middle

NTP     Network Time Protocol [[RFC5905](#)]

NTS     Network Time Security

TESLA   Timed Efficient Stream Loss-Tolerant Authentication

MAC    Message Authentication Code

HMAC   Keyed-Hash Message Authentication Code

### **4. Overview of NTS-Secured NTP**

#### **4.1. Symmetric and Client/Server Mode**

The server does not keep a state of the client. NTS applies X.509 certificates to verify the authenticity of the time server and to exchange a symmetric key, the so-called cookie. The "association" and "cookie" message exchanges are utilized for this. In subsequent "unicast time synchronization" message exchanges, the cookie is then used to protect authenticity and integrity of NTP unicast time synchronization packets. This is achieved by a MAC attached to each time synchronization packet.

#### **4.2. Broadcast Mode**

After the client has accomplished the necessary initial time synchronization via client-server mode, a "broadcast parameter" message exchange is utilized to communicate the necessary broadcast parameters to the client. Subsequently, "broadcast time synchronization" message exchanges are utilized in combination with optional "broadcast keycheck" exchanges to protect authenticity and integrity of NTP broadcast time synchronization packets. This is also achieved by MACs.

### **5. Protocol Sequence**

#### **5.1. The Client**

##### **5.1.1. 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. It MUST keep the transmitted nonce as well as the values for the version number and algorithms available for later checks.
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 client checks that the message contains a conforming version number.
  - \* It checks that the nonce sent back by the server matches the one transmitted in `client_assoc`,
  - \* It also verifies that the server has chosen the encryption and hash algorithms from its proposal sent in the `client_assoc` message and that this proposal was not altered.
  - \* Furthermore, it performs authenticity checks on the certificate chain and the signature.

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

Discussion:

Note that by performing the above message exchange and checks, 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 [RFC 7384](#) [[RFC7384](#)] depends on the chosen trust anchor.

3. Next it sends a `client_cook` message to the server. The client MUST save the included nonce until the reply has been processed.
4. It awaits a reply in the form of a `server_cook` message; upon receipt it executes the following actions:
  - \* It verifies that the received version number matches the one negotiated beforehand.
  - \* It verifies the signature using the server's public key. The signature has to authenticate the encrypted data.
  - \* It decrypts the encrypted data with its own private key.
  - \* It checks that the decrypted message is of the expected format: the concatenation of a 128 bit nonce and a 128 bit cookie.





- \* It verifies that the received nonce matches the nonce sent in the `client_cook` message.

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

5. The client sends a `time_request` message to the server. The client **MUST** save the included nonce and the `transmit_timestamp` (from the time synchronization data) as a correlated pair for later verification steps.
6. It awaits a reply in the form of a `time_response` message. Upon receipt, it checks:
  - \* that the transmitted version number matches the one negotiated previously,
  - \* that the transmitted nonce belongs to a previous `time_request` message,
  - \* that the `transmit_timestamp` in that `time_request` message matches the corresponding time stamp from the synchronization data received in the `time_response`, and
  - \* that the appended MAC verifies the received synchronization data, version number and nonce.

If at least one of the first three checks fails (i.e. if the version number does not match, if the client has never used the nonce transmitted in the `time_response` message, or if it has used the nonce with initial time synchronization data different from that in the response), then 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 repeating the exchange of `time_request` and `time_response` messages.

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

#### **5.1.2. 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 with 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. It MUST remember the transmitted values for the nonce, the version number and the signature algorithm.
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 specified for a `server_bpar` message.
  - \* The message must contain a nonce belonging to a `client_bpar` message that the client has previously sent.
  - \* Verification of the message's signature.

If any information is missing or if the server's signature cannot be verified, the client MUST abort the broadcast run. If all checks are successful, the client MUST remember all the broadcast parameters received for later checks.

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 (packet timeliness). This is achieved via a combination of checks. First, the disclosure schedule is used, which requires loose time synchronization. If this is successful, the client obtains a stronger guarantee via a key check exchange: it sends a `client_keycheck` message and waits for the appropriate response. Note that it needs to memorize the nonce and the time interval number that it sends as a correlated pair. For more detail on both of the mentioned timeliness checks, see [[I-D.ietf-ntp-network-time-security](#)]. If its timeliness is 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 also be verified.
  2. The client checks that it does not already know the disclosed key. Otherwise, the client SHOULD discard the packet to avoid a buffer overrun. If verified, the client ensures 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 shown. If it is falsified, the client MUST discard the packet.



3. If the disclosed key is legitimate, then the client verifies the authenticity of any packet that it has 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, then 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. Also, the client MUST re-initialize the broadcast sequence with a "client\_bpar" message if the one-way key chain expires, which it can check via the disclosure schedule.

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

The client MUST restart the broadcast sequence with a client\_bpar message ([\[I-D.ietf-ntp-network-time-security\]](#)) if the one-way key chain expires.

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

## **5.2. The Server**

### **5.2.1. The Server in Unicast Mode**

To support unicast mode, the server MUST be ready to perform the following actions:

- o 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 [\[I-D.ietf-ntp-network-time-security\]](#).
- o Upon receipt of a client\_cook message, the server checks whether it supports the given cryptographic algorithms. It then calculates the cookie according to the formula given in [Section 4.1](#). With this, it MUST construct a server\_cook message as described in [\[I-D.ietf-ntp-network-time-security\]](#).
- o 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 [\[I-D.ietf-ntp-network-time-security\]](#).

The server MUST refresh its server seed periodically (see [\[I-D.ietf-ntp-network-time-security\]](#)).



### **5.2.2. The Server in Broadcast Mode**

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

- o Upon receipt of a client\_bpar message, the server constructs and sends a server\_bpar message as described in [\[I-D.ietf-ntp-network-time-security\]](#).
- o Upon receipt of a client\_keycheck message, the server looks up whether it has already disclosed the key associated with the interval number transmitted in that message. If it has not disclosed it, it constructs and sends the appropriate server\_keycheck message as described in [\[I-D.ietf-ntp-network-time-security\]](#). For more details, see also [\[I-D.ietf-ntp-network-time-security\]](#).
- o The server follows the TESLA protocol in all other aspects, by regularly sending server\_broad messages as described in [\[I-D.ietf-ntp-network-time-security\]](#), 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.

## **6. Implementation Notes: ASN.1 Structures and Use of the CMS**

This section presents some hints about the structures of the communication packets for the different message types when one wishes to implement NTS for NTP. See document [\[I-D.ietf-ntp-cms-for-nts-message\]](#) for descriptions of the archetypes for CMS structures as well as for the ASN.1 structures that are referenced here.

### **6.1. Unicast Messages**

#### **6.1.1. Association Messages**

##### **6.1.1.1. Message Type: "client\_assoc"**

This message is realized as an NTP packet with an extension field which holds an "NTS-Plain" archetype CMS structure. This structure contains in its core an NTS message object of the type "ClientAssociationData", which holds all the data necessary for the NTS security mechanisms.





#### **6.1.1.2. Message Type: "server\_assoc"**

Like "client\_assoc", this message is realized as an NTP packet with an extension field which holds an "NTS-Plain" archetype CMS structure. This structure contains in its core an NTS message object of the type "ServerAssociationData". The latter holds all the data necessary for NTS.

### **6.1.2. Cookie Messages**

#### **6.1.2.1. Message Type: "client\_cook"**

This message type is realized as an NTP packet with an extension field which holds a CMS structure of archetype "NTS-Certified", containing in its core an NTS message object of the type "ClientCookieData". The latter holds all the data necessary for the NTS security mechanisms.

#### **6.1.2.2. Message Type: "server\_cook"**

This message type is realized as an NTP packet with an extension field containing a CMS structure of archetype "NTS-Signed-and-Encrypted". The NTS message object in that structure is a "ServerCookieData" object, which holds all data required by NTS for this message type.

### **6.1.3. Time Synchronization Messages**

#### **6.1.3.1. Message Type: "time\_request"**

This message type 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" (packed in a CMS structure of archetype "NTS-Plain"), whose structure is as follows:

#### **6.1.3.2. Message Type: "time\_response"**

This message is also realized as an NTP packet with regular NTP time synchronization data. The packet also has an extension field which contains an ASN.1 object of type "TimeResponseSecurityData". Finally, this NTP packet has a MAC field which contains a Message Authentication Code generated over the whole packet (including the extension field).



## **6.2. Broadcast Messages**

### **6.2.1. Broadcast Parameter Messages**

#### **6.2.1.1. Message Type: "client\_bpar"**

This first broadcast message is realized as an NTP packet which is empty except for an extension field which contains an ASN.1 object of type "BroadcastParameterRequest" (packed in a CMS structure of archetype "CMS-Plain"). This is sufficient to transport all data specified by NTS.

#### **6.2.1.2. Message Type: "server\_bpar"**

This message type is realized as an NTP packet whose extension field carries the necessary CMS structure (archetype: "NTS-Signed"). The NTS message object in this case is an ASN.1 object of type "BroadcastParameterResponse".

### **6.2.2. Broadcast Time Synchronization Message**

#### **6.2.2.1. Message Type: "server\_broad"**

This message's 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" (packed in a CMS structure with archetype "NTS-Plain"). In addition to all this, this packet has a MAC field which contains a Message Authentication Code generated over the whole packet (including the extension field).

### **6.2.3. Broadcast Keycheck**

#### **6.2.3.1. Message Type: "client\_keycheck"**

This message is realized as an NTP packet with an extension field, which transports a CMS structure of archetype "NTS-Plain", containing an ASN.1 object of type "ClientKeyCheckSecurityData".

#### **6.2.3.2. Message Type: "server\_keycheck"**

This message is also realized as an NTP packet with an extension field, which contains an ASN.1 object of type "ServerKeyCheckSecurityData" (packed in a CMS structure of archetype "NTS-Plain"). Additionally, this NTP packet has a MAC field which contains a Message Authentication Code generated over the whole packet (including the extension field).



## **7. Security Considerations**

### **7.1. Usage of NTP Pools**

The certification-based authentication scheme described in [[I-D.ietf-ntp-network-time-security](#)] is not applicable to the concept of NTP pools. Therefore, NTS is unable to provide secure usage of NTP pools.

### **7.2. Server Seed Lifetime**

tbd

### **7.3. Supported Hash Algorithms**

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

- o it MUST NOT include SHA-1 or weaker algorithms,
- o it MUST include SHA-256 or stronger algorithms.

## **8. Acknowledgements**

The authors would like to thank Russ Housley, Steven Bellovin, 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.

## **9. References**

### **9.1. Normative References**

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



## **9.2. Informative References**

- [I-D.ietf-ntp-cms-for-nts-message]  
Sibold, D., Roettger, S., Teichel, K., and R. Housley,  
"Protecting Network Time Security Messages with the  
Cryptographic Message Syntax (CMS)", [draft-ietf-ntp-cms-  
for-nts-message-01](#) (work in progress), January 2015.
- [I-D.ietf-ntp-network-time-security]  
Sibold, D., Roettger, S., and K. Teichel, "Network Time  
Security", [draft-ietf-ntp-network-time-security-07](#) (work  
in progress), March 2015.
- [RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in  
Packet Switched Networks", [RFC 7384](#), October 2014.

## **Appendix A. Flow Diagrams of Client Behaviour**





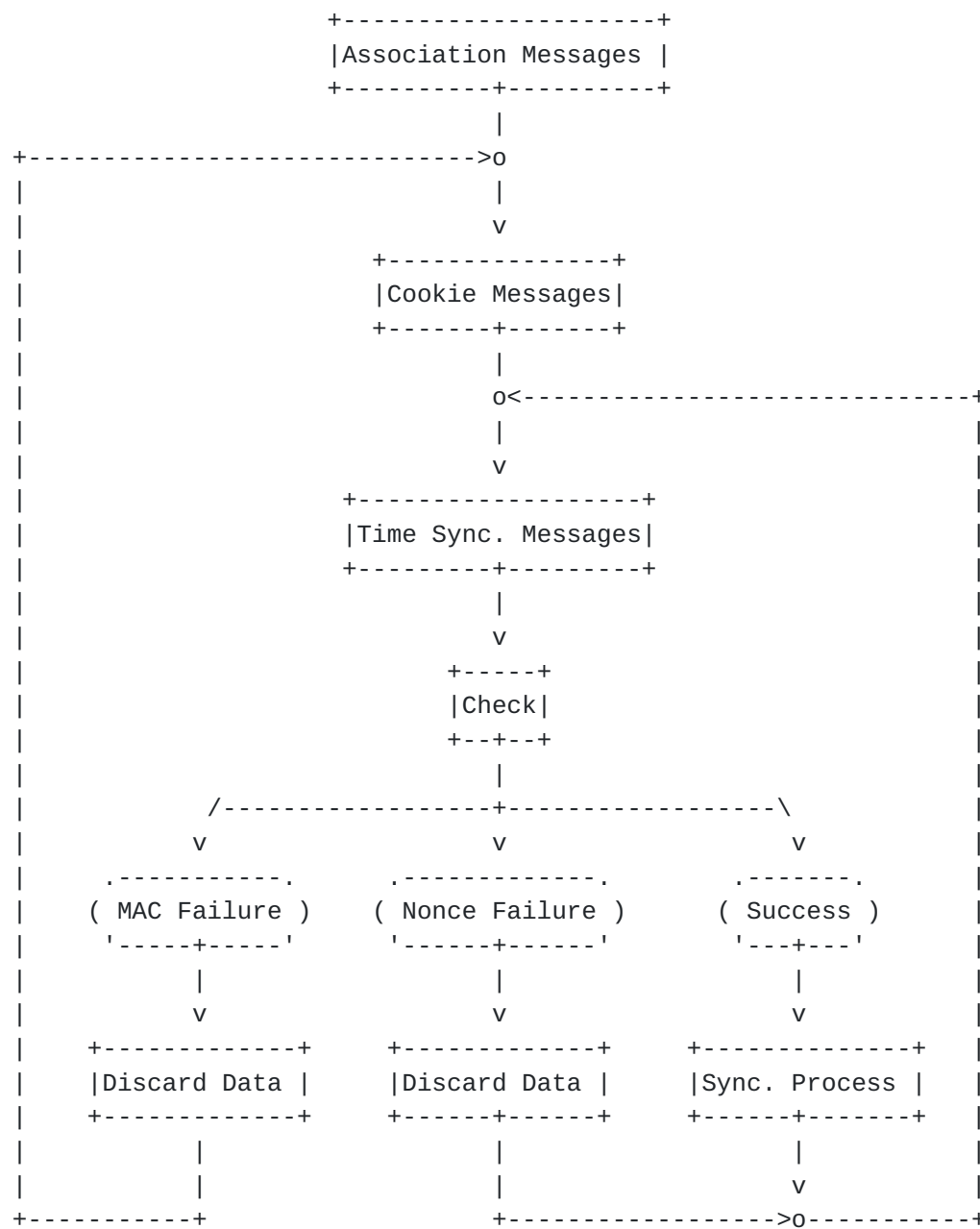


Figure 1: The client's behavior in NTS unicast mode.



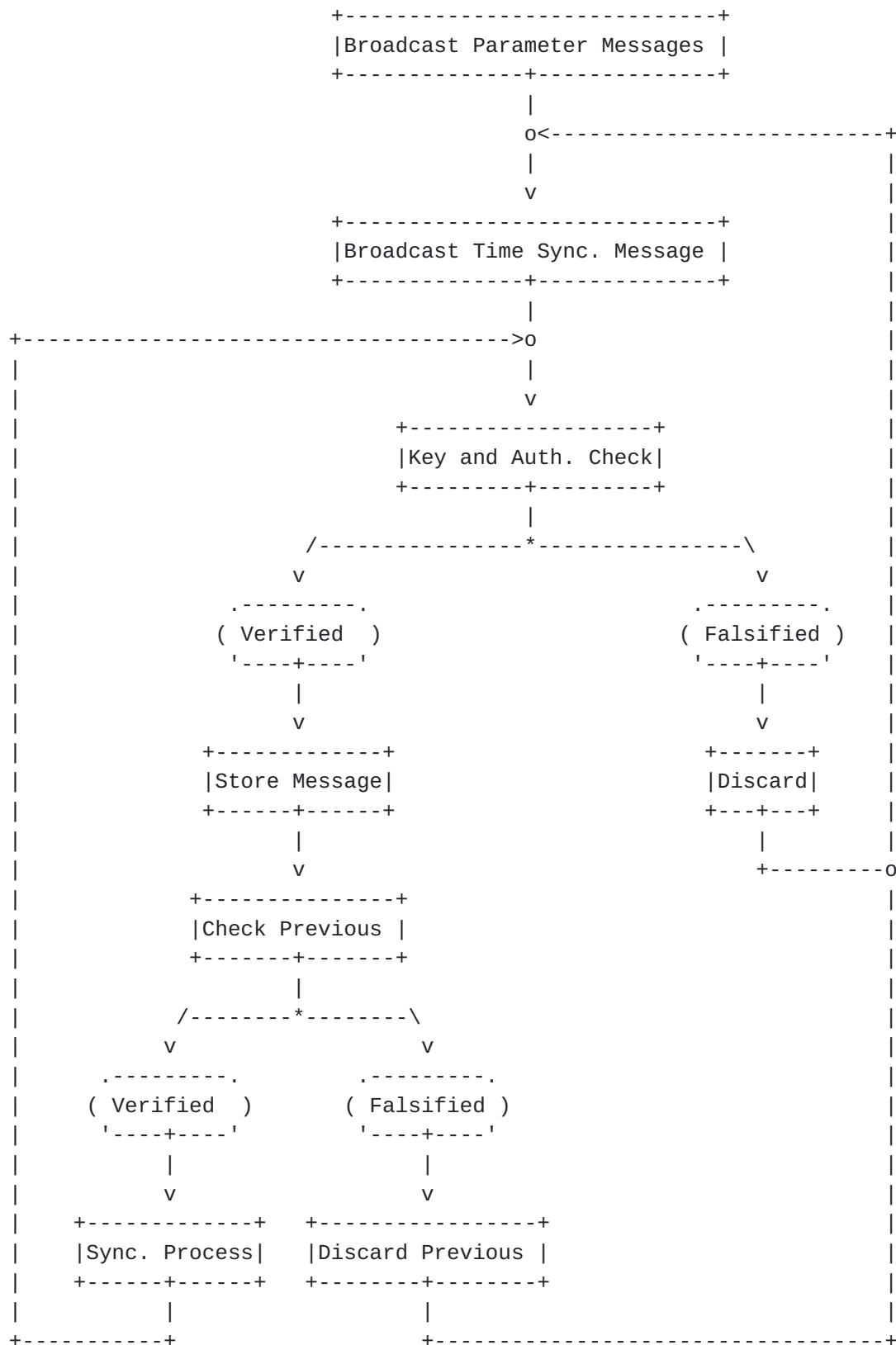


Figure 2: The client's behaviour in NTS broadcast mode.



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  
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

