

NTP Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: August 25, 2013

D. Sibold  
PTB  
S. Roettger  
TU-BS  
February 23, 2013

Network Time Protocol: autokey Version 2 Specification  
draft-sibold-autokey-02

## Abstract

This document describes a security protocol that enables authenticated time synchronization using Network Time Protocol (NTP). Autokey Version 2 obsoletes NTP autokey protocol [RFC 5906](#) [[RFC5906](#)] which suffers from various security vulnerabilities. Its design considers the special requirements that are related to the task of precise timekeeping.

## Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [[RFC2119](#)].

## Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 25, 2013.

## Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

Internet-Draft

NTP autokey V2

February 2013

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>	<a href="#">Introduction . . . . .</a>	<a href="#">2</a>
<a href="#">1.1.</a>	<a href="#">Differences from the original autokey . . . . .</a>	<a href="#">3</a>
<a href="#">2.</a>	<a href="#">Security Threats . . . . .</a>	<a href="#">3</a>
<a href="#">3.</a>	<a href="#">Objectives . . . . .</a>	<a href="#">3</a>
<a href="#">4.</a>	<a href="#">Terms and abbreviations . . . . .</a>	<a href="#">4</a>
<a href="#">5.</a>	<a href="#">Autokey Overview . . . . .</a>	<a href="#">4</a>
<a href="#">5.1.</a>	<a href="#">Symmetric and Client/Server Mode . . . . .</a>	<a href="#">4</a>
<a href="#">5.2.</a>	<a href="#">Broadcast Mode . . . . .</a>	<a href="#">4</a>
<a href="#">6.</a>	<a href="#">Protocol Sequence . . . . .</a>	<a href="#">5</a>
<a href="#">6.1.</a>	<a href="#">Association Message . . . . .</a>	<a href="#">5</a>
<a href="#">6.2.</a>	<a href="#">Certificate Message . . . . .</a>	<a href="#">5</a>
<a href="#">6.3.</a>	<a href="#">Cookie Message . . . . .</a>	<a href="#">6</a>
<a href="#">6.4.</a>	<a href="#">Broadcast Parameter Message . . . . .</a>	<a href="#">6</a>
<a href="#">6.5.</a>	<a href="#">Time Request Message . . . . .</a>	<a href="#">6</a>
<a href="#">6.6.</a>	<a href="#">Broadcast Message . . . . .</a>	<a href="#">6</a>
<a href="#">7.</a>	<a href="#">Hash algorithms and MAC generation . . . . .</a>	<a href="#">7</a>
<a href="#">7.1.</a>	<a href="#">Hash algorithms . . . . .</a>	<a href="#">7</a>
<a href="#">7.2.</a>	<a href="#">MAC Calculation . . . . .</a>	<a href="#">7</a>
<a href="#">8.</a>	<a href="#">Server Seed Considerations . . . . .</a>	<a href="#">8</a>
<a href="#">8.1.</a>	<a href="#">Server Seed algorithm . . . . .</a>	<a href="#">8</a>
<a href="#">8.2.</a>	<a href="#">Server Seed Live Time . . . . .</a>	<a href="#">8</a>
<a href="#">9.</a>	<a href="#">IANA Considerations . . . . .</a>	<a href="#">8</a>
<a href="#">10.</a>	<a href="#">Security Considerations . . . . .</a>	<a href="#">8</a>
<a href="#">11.</a>	<a href="#">Acknowledgements . . . . .</a>	<a href="#">8</a>
<a href="#">12.</a>	<a href="#">References . . . . .</a>	<a href="#">8</a>
<a href="#">12.1.</a>	<a href="#">Normative References . . . . .</a>	<a href="#">8</a>
<a href="#">12.2.</a>	<a href="#">Informative References . . . . .</a>	<a href="#">9</a>
<a href="#">Appendix A.</a>	<a href="#">TICTOC Security Requirements . . . . .</a>	<a href="#">9</a>
<a href="#">Appendix B.</a>	<a href="#">Broadcast Mode . . . . .</a>	<a href="#">10</a>
	<a href="#">Authors' Addresses . . . . .</a>	<a href="#">10</a>

## 1. Introduction

In NTP [[RFC5905](#)] the autokey protocol [[RFC5906](#)] was introduced to provide authenticity to NTP servers and to ensure integrity of time synchronization. It is designed to meet the specific communication requirements of precise timekeeping and therefore does not compromise timekeeping precision.

This document focuses on a new definition of the autokey protocol for NTP, autokey version 2. The necessity to renew the autokey specification arises from various severe security vulnerabilities that have been found in a thorough analysis of the protocol [[Roettger](#)]. The new specification is based on the same assumptions as the original autokey specification. In particular, the prerequisite is that precise timekeeping can only be accomplished with stateless time synchronization communication, which excludes standard security protocols like IPsec or TLS. 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].

### 1.1. Differences from the original autokey

Autokey version 2 is a major redraft of the original autokey specification. It is intended to mitigate security vulnerabilities of the original specification and it is based on the suggestions in the analysis of Roettger [[Roettger](#)]. The major changes are:

- o The bit length of server seed and cookie has been increased.
- o The IP addresses of the synchronization partners in the calculation of the cookie have been replaced by the hash value of the client's public key.
- o The identity schemes for the verification of the NTP server authenticity have been replaced by a hierarchical public key infrastructure (PKI) based on X.509 certificates.

## [2.](#) Security Threats

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

## [3.](#) Objectives

The objectives of the autokey specifications are as follows:

- o Authenticity: Autokey enables the client to authenticate its NTP server or peer.
- o Integrity: Autokey protects the integrity of time synchronization packets via a message authentication code (MAC).
- o Confidentiality: Autokey does not provide confidentiality

protection of the NTP packets.

- o Modes of operation: All operational modes of NTP are supported (client server, symmetric, broadcast).
- o Hybrid mode: Both secure and insecure communication modes are possible for NTP servers and clients, respectively.
- o Compatibility:
  - \* Interoperation with autokey version 1 is not given.
  - \* NTP associations without authentication shall not be affected.
  - \* An NTP server that does not support autokey version 2 shall not be affected by autokey version 2 authentication requests.

## [4.](#) Terms and abbreviations

- o Throughout this document the term "autokey" refers to autokey version 2.

- o TESLA: Time efficient stream loss-tolerant authentication

## [5.](#) Autokey Overview

### [5.1.](#) Symmetric and Client/Server Mode

Authenticity and integrity of the NTP packets are 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:

$$\text{cookie} = \text{MSB}_{128} (\text{H}(\text{server seed} || \text{H}(\text{public key of client}))),$$

where `||` indicates concatenation and in which H is a hash algorithm. The function MSB<sub>128</sub> cuts off the 128 most significant bits of the result of the hash function. The server seed is a 128 bit random value of the server, which has to be kept secret. The cookie thus never changes. The server seed has to be refreshed periodically. 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 6.5](#)).

### [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 [[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 interval. The client verifies the MAC by buffering the packet until the disclosure of the key in the next 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.

## [6.](#) Protocol Sequence

### [6.1.](#) Association Message

The protocol sequence starts with the association message, in which the client sends an NTP packet with an extension field of type association. It contains the hostname of the client and a status word which contains the algorithms used for the signatures and the status of the connection. The response contains the hostname of the server and the algorithms for the signatures. The server notifies the cryptographic hash algorithms which it supports.

### [6.2.](#) Certificate Message

In this step, the client receives the certification chain up to the trusted authority (TA). To this end, the client requests the certificate for the subject name (hostname) of the NTP server. The response contains the certificate with the issuer name. If the issuer name is different from the subject name, the client requests the certificate for the issuer. This continues until it receives a certificate which is issued by a TA. The client recognizes the TA because it has a list of certificates which are accepted as TAs. The client has to check that each issuer is authorized to issue new certificates. To this end, the certificates have to include the X.509v3 extension field "CA:TRUE". 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 validate the authenticity of its

NTP-server only. It does not recursively validate the authenticity of each NTP server on the time synchronization chain. But each NTP server on the time synchronization chain validates the NTP server to which it is synchronized. This conforms to the recursive authentication requirement in the TICTOC security requirements [[I-D.ietf-tictoc-security-requirements](#)].

### [6.3.](#) Cookie Message

The client requests a cookie from the server. It selects a hash algorithm from the list of algorithms supported by the server. The request includes its public key and the selected hash algorithm. The hash of the public key is used by the server to calculate the cookie (see [Section 5.1](#)). The response of the server contains the cookie encrypted with the public key.

### [6.4.](#) Broadcast Parameter Message

In the broadcast mode the client requests the following information from the server:

- o the last key of the one-way key chain,
- o the disclosure schedule of the following keys. This contains:
  - \* time interval duration, time at which the next time interval will start and its associated index,
  - \* key disclosure delay (number of time intervals for which a key is valid).

The server will sign all transmitted properties so that the client is able to verify their authenticity. For this packet exchange a new extension field "broadcast parameters" is used. The client synchronizes its time with the server in the client server mode and saves an upper bound of its time offset with respect to the time of the server. See [Appendix B](#) for more details.

### [6.5.](#) Time Request Message

The client request includes a new extension field "time request" which contains the hash of its public key. The server needs the hash of the public key to recalculate the cookie for the client. The response is a normal NTP packet without extension field. It contains a MAC.

### [6.6.](#) Broadcast Message

Internet-Draft

NTP autokey V2

February 2013

The NTP broadcast packet includes a new extension field "broadcast message" which contains the disclosed key of the previous disclosure interval (current time interval minus disclosure delay). The NTP packet is appended by a MAC, calculated with the key for the current time interval. When a client receives a broadcast message it has to perform the following tests:

- o Proof that the MAC is based on a key that is not yet disclosed. If verified the packet will be buffered for later authentication otherwise it has to be discarded.
- o The client checks whether it already knows the disclosed key. If not, the client verifies its legitimacy. If falsified the packet has to be discarded.
- o 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. 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 [Appendix B](#) or [\[RFC4082\]](#) for a detailed description of the packet verification process.

## [7.](#) Hash algorithms and MAC generation

### [7.1.](#) Hash algorithms

Hash algorithms are used at different points: calculation of the cookie and the MAC, and hashing of the public key. The client selects the hash algorithm from the list of hash algorithms which are supported by the server. This list is notified during the association message exchange ([Section 6.1](#)). The selected algorithm is used for all hashing processes in the protocol.

In the broadcast mode hash algorithm are used as pseudo random function to construct the one-way key chain.

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



- o it MUST NOT contain the MD5 or weaker algorithms,
- o it MUST include SHA-256 or stronger algorithms.

## [7.2.](#) 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 7.1](#)).

Sibold & Roettger

Expires August 25, 2013

[Page 7]

---

Internet-Draft

NTP autokey V2

February 2013

## [8.](#) Server Seed Considerations

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

### [8.1.](#) Server Seed algorithm

### [8.2.](#) Server Seed Live Time

## [9.](#) IANA Considerations

This document makes no request of IANA.

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

## [10.](#) Security Considerations

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:

- o The validity of the certificates is preconditioned. Usually this will be the case in corporation networks.
- o The client ensures that the certificates are not revoked. To this end, the client uses the Online Certificate Status Protocol (OCSP) defined in [[RFC6277](#)].

- o 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.](#) Acknowledgements

## [12.](#) References

### [12.1.](#) Normative References

- [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.

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

### [12.2.](#) Informative References

- [I-D.ietf-tictoc-security-requirements] Mizrahi, T., "Security Requirements of Time Synchronization Protocols in Packet Switched Networks", Internet-Draft [draft-ietf-tictoc-security-requirements-04](#), February 2013.
- [RFC2104] Krawczyk, H., Bellare, M. and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", [RFC 2104](#), February 1997.
- [RFC4082] Perrig, A., Song, D., Canetti, R., Tygar, J.D. 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.

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

## [Appendix A](#). TICTOC Security Requirements

The following table compares the autokey specifications against the TICTOC security requirements [[I-D.ietf-tictoc-security-requirements](#)].

Section	Requirement from I-D tictoc security-requirements-04	Requirement level	Autokey V2
5.1	Clock Identity Authentication and Authorization	MUST	OK
5.1.1	Authentication and Authorization of Masters	MUST	OK
5.1.2	Recursive Authentication and Authorization of Masters (Chain of Trust)	MUST	OK

5.1.3	Authentication and Authorization of Slaves	MAY	-
5.2	Integrity protection.	MUST	OK
5.3	Protection against DoS attacks	SHOULD	-
5.4	Replay protection	MUST	OK (NTP)
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	-
5.8	Protection against Packet Delay and Interception Attacks	SHOULD	-
5.9.1	Secure mode	MUST	OK (NTP)
5.9.2	Hybrid mode	MAY	OK (NTP)

Comparison between TICTOC security requirements and autokey.

## [Appendix B.](#) Broadcast Mode

### Authors' Addresses

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

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

Sibold & Roettger

Expires August 25, 2013

[Page 10]

Internet-Draft

NTP autokey V2

February 2013

Stephen Roettger

Technische Universitaet Braunschweig

Email: [stephen.roettger@googlemail.com](mailto:stephen.roettger@googlemail.com)

