[MS-SNTP]:
Network Time Protocol (NTP) Authentication Extensions

Revision Summary

Date	Revision History	Revision Class	Comments
12/18/2006	0.1		MCPP Milestone 2 Initial Availability
03/02/2007	1.0		MCPP Milestone 2
04/03/2007	1.1		Monthly release
05/11/2007	1.2		Monthly release
06/01/2007	2.0	Major	Updated and revised the technical content.

Date	Revision History	Revision Class	Comments
07/03/2007	2.1	Minor	Updated the technical content.
07/20/2007	3.0	Major	Rewrite to include dependencies on Netlogon.
08/10/2007	3.0.1	Editorial	Revised and edited the technical content.
09/28/2007	3.0.2	Editorial	Revised and edited the technical content.
10/23/2007	3.0.3	Editorial	Revised and edited the technical content.
11/30/2007	3.1	Minor	Replaced reference for SNTP.
01/25/2008	3.2	Minor	Updated the technical content.

Table of Contents

1 Introduction

This document specifies an authentication extension to the Network Time Protocol (NTP) version 3, as specified in [RFC1305], and the Simple Network Time Protocol (SNTP) version 4, as specified in [RFC2030].

[RFC1305] Appendix C describes a mechanism similar to the authentication extension documented here. The extension documented here provides for better security by using a stronger checksum algorithm, and by using keying material that is more convenient for Windows systems joined to a Windows **domain**.

SNTP, as specified in [RFC2030], provides a simplified version of NTP. [RFC2030] section 4 refers to the **Authenticator** field of [RFC1305] Appendix A; the extension documented here replaces that referral, and specifies the way in which an SNTP message exchange can be secured.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

Domain
Domain Controller (DC)
Domain Controller Locator
Domain Object
Relative Identifier (RID)
Security Identifier (SID)
Time Service
Trusted Domain Object (TDO)

The following terms are specific to this document:

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[MS-ADTS] Microsoft Corporation, "Active Directory Technical Specification", June 2007.

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[MS-NLMP] Microsoft Corporation, "NT LAN Manager (NTLM) Authentication Protocol Specification", June 2007.

[MS-NRPC] Microsoft Corporation, "Netlogon Remote Protocol Specification", March 2007.

[MS-SECO] Microsoft Corporation, "Windows Security Overview", January 2007.

[MS-W32T] Microsoft Corporation, "W32Time Remote Protocol Specification", September 2007.

[RFC1305] Mills, D. L., "Network Time Protocol (Version 3) Specification, Implementation and Analysis", RFC 1305, March 1992, http://www.ietf.org/rfc/rfc1305.txt

[RFC2030] Mills, D., "Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI", RFC 2030, October 1996, http://www.faqs.org/rfcs/rfc2030.html

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

1.2.2 Informative References

[MD5Collision] Klima, V., "Tunnels in Hash Functions: MD5 Collisions Within a Minute", March 2006, http://eprint.iacr.org/2006/105.pdf

1.3 Protocol Overview (Synopsis)

1.3.1 Background

NTP Version 3, as specified in [RFC1305], is used to synchronize time between a client and a server. The client sends a request for time synchronization information to the server, and the server replies with the information requested.

The NTP authentication extension in [RFC1305] Appendix C is designed to protect against unauthorized responses by using a crypto-checksum computed by the server and checked by the client, using a predefined encryption algorithm with cryptographic keys indexed by a key identifier included in the message.

However, no provisions exist in the RFC for how to securely distribute and maintain cryptographic keys or the key identifier. These issues are the responsibility of the implementers of [RFC1305]. The extension described in this document explains how this is done within a Windows domain.

SNTP, as specified in [RFC2030], is a simplified version of NTP that provides a coarser granularity of clock synchronization. [RFC2030] itself does not provide for authentication, and refers to the relevant authentication fields of [RFC1305] Appendix C. The NTP Authentication Extensions specified in this document provide a way to add authentication to SNTP to the same degree as provided in NTP.

1.3.2 Extensions

The NTP Authentication Extensions use the Netlogon Remote Protocol (as specified in [MS-NRPC]) in a Windows domain environment for authentication between a client and server that are capable of establishing a secure connection.

As specified in [MS-NRPC] section 3.5.4.2, the client uses the Netlogon **domain controller locator** service to find a **domain controller** that is a time source. The client and domain controller must have pre-established trusted account information in the Windows domain. The pre-established trusted account information is used to establish cryptographic keys and associated key identifiers for NTP authentication between the client and server.

The client sends an authentication request for time-synchronization information with the key identifier. The server constructs the response with the time information requested. The server uses Netlogon message protection methods, as specified in [MS-NRPC] section 3.5.4.6, to compute the crypto-checksum of the message. The server replies in an authentication response with the time information requested and the computed crypto-checksum. The client authenticates the authentication response by computing and matching the checksum using the Netlogon message

protection methods, as specified in [MS-NRPC] section 3.5.4.6. The client accepts only authenticated responses. The sequence diagram is as described in sections 3.1.5 and 3.2.5.

This document specifies an authentication mechanism that differs from the interim NTP authentication extension defined in [RFC1305] Appendix C as follows:

- Uses a modified authenticator format and semantics, as specified in section 2.2.
- Uses a modified algorithm to compute the checksum, as specified in sections 3.1.5 and 3.2.5.
- Specifies how cryptographic keys are established and obtained in Windows domains, as specified in section 1.5.
- Specifies how keys are identified by a client that is synchronizing time using the authentication extension against a Windows domain controller, as specified in section 1.5.
- Specifies how key identifiers must be interpreted by a server that is providing time synchronization for a Windows client using the authentication extension, as specified in section 1.5.
- Uses modified authentication procedures, as specified in sections 3.1.5, 3.1.6, and 3.2.5.

The NTP Authentication Extensions also apply to SNTP, as specified in [RFC2030]. The NTP Authentication Extensions specified in this document override the statements in [RFC2030] section 4 pertaining to the **Authenticator** field, and instead specify how the field is to be used and interpreted.

The NTP Authentication Extensions are defined only for the following NTP and SNTP association modes (as specified in [RFC1305] sections 3.2.1 and 3.3): client and server. The client/server mode refers to the roles within the context of the NTP protocol, as opposed to roles within a network system.

As well as extending [RFC1305], the NTP Authentication Extensions apply directly to SNTP, as specified in [RFC2030]. For simplicity, only the terms NTP and [RFC1305] are generally used in the main body of this document. All references to NTP and [RFC1305] must be taken to apply equally to SNTP and [RFC2030] unless the text clearly specifies otherwise.<1>

1.4 Relationship to Other Protocols

For locating a domain controller that is a time source, the extension relies on the Netlogon domain controller locator service, as specified in [MS-NRPC] section 3.5.4.2. For authenticating message exchanges, the extension relies on NetLogon message protection methods, as specified in [MS-NRPC] section 3.5.4.6.

No known higher-layer protocols depend on the existence of this extension.

The W32Time Remote Protocol, as specified in [MS-W32T], is an RPC interface protocol used for controlling and monitoring the Windows implementation of NTP.

1.5 Prerequisites/Preconditions

The NTP Authentication Extensions specify NTP authentication in a Windows domain environment. The extensions, which rely on the Netlogon Remote Protocol (as specified in [MS-NRPC]), assume that a secure connection can be established between the client and a domain controller.

1.5.1 Time Source Discovery and Selection

The client must have a way of locating a time source that is a domain controller and that can establish a secure connection with the client.

As specified in [MS-NRPC] section 3.5.4.2, Windows clients use the DsrGetDcName method in the Netlogon domain controller locator service to find their time sources. Each Windows domain controller configured to be a time source must set its domain control information flags with the appropriate **time service** flags, as specified in [MS-NRPC] section 3.5.4.2.

1.5.2 Key Establishment and Identifying Keys

The client and server must have an implementation-specific mechanism for mutually establishing cryptographic keys and associating them with key identifiers. The client and server must have an implementation-specific mechanism for accessing the cryptographic keys and the associated key identifiers. There must be at least one association of key identifier and cryptographic key, and at most two.

In the Windows implementation, authentication makes use of the NetLogon message protection methods, as specified in [MS-NRPC] section 3.5.4.6, as follows:

- The NetrLogonGetTimeServiceParentDomain method (as specified in [MS-NRPC] section **3.5.4.6.4**) is used to obtain a trusted domain in which a secure connection can be established between the client and server.
- The NetrLogonGetTrustRid method (as specified in [MS-NRPC] section **3.5.4.6.1**) is used to obtain the **relative identifier (RID)** of a trusted account whose passwords are used to establish a secure connection between the client and server. The account must be associated with a pair of keys, which are the old and new passwords for the account.
- The NTOWFv1 (as specified in [MS-NLMP] section 3.3.1) of the old and new passwords are used as the cryptographic keys for NTP authentication.
- The least significant 31 bits of the NTP authentication key identifier are set to the least significant 31 bits of the RID of the trusted account.
- The most significant bit of the key identifier is a 1-bit key selector that indexes the pair of keys associated with the account. The bit is set to 0 to identify the cryptographic key as the NTOWFv1 of the new password. The bit is set to 1 to identify the cryptographic key as the NTOWFv1 of the old password.

1.5.3 Client Configuration for Authentication

A client MAY need to be manually configured to participate in the NTP Authentication Extensions detailed in this specification.

On a Windows machine that is joined to a Windows domain, the client is configured to use this authentication extension by default.

1.6 Applicability Statement

The NTP Authentication Extensions are designed primarily for authenticated time synchronization in Windows domains between Windows clients and Windows domain controllers.

A Windows client joined to a Windows domain will use the NTP Authentication Extensions by default.

1.7 Versioning and Capability Negotiation

The NTP Authentication Extensions incorporate implicit capability negotiation based on the NTP message length, as described in sections 3.1.5, 3.1.6, and 3.2.5.

NTP Version 3, as specified in [RFC1305], and SNTP Version 4, as specified in [RFC2030], apply to this extension.

1.8 Vendor-Extensible Fields

The NTP Authentication Extensions do not define any vendor-extensible fields. They do, however, redefine the **Authenticator** field of [RFC1305] and [RFC2030] from 96 to 160 bits, as specified in section 2.2.

1.9 Standards Assignments

The NTP Authentication Extensions do not use any additional standards assignments other than the ones in the base protocols.

2 Messages

This section describes how the **Authenticator** field is encapsulated on the wire.

2.1 Transport

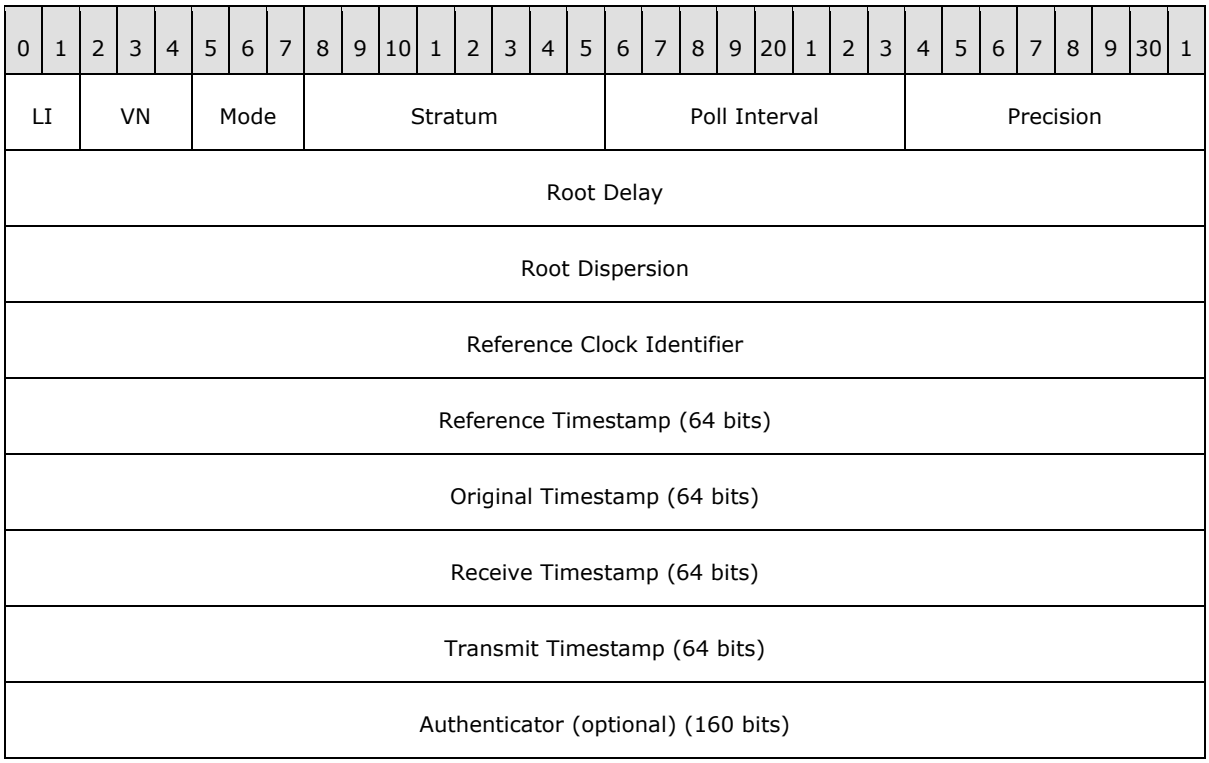
All NTP messages are transported through the User Datagram Protocol (UDP) in the NTP data format, as specified in [RFC1305] Appendix A.

NTP messages do not include a message length field. The NTP message length is calculated based on the payload length in the UDP header and excludes the length of the UDP header.

Security parameters used in the NTP Authentication Extensions are cryptographic keys with their associated key identifiers, as described in section 1.5.2.

2.2 Message Syntax

The complete format of an NTP message is as follows. The format is identical to the NTP Version 3 message, as specified in [RFC1305] Appendix A, with the exception of the optional **Authenticator** field. This document defines no fields other than the optional **Authenticator** field. For more information on NTP Version 3 message formats, see [RFC1305] Appendix A.



The Windows implementation specifies one additional ASCII identifier "LOCL" for the reference-clock identifier field defined in [RFC1305] Appendix A. If a primary server has Stratum set to 1, the server MAY use the reference-clock identifier "LOCL". This identifier indicates that the primary server is responding to requests using its own local computer clock without being synchronized to other external clocks.

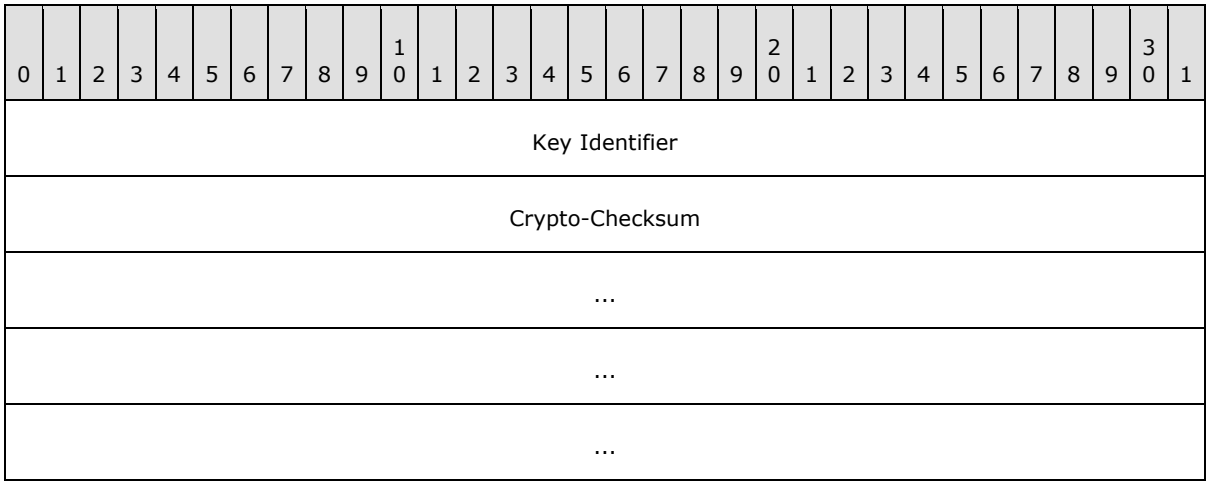
The optional **Authenticator** field used in the NTP authentication extension was originally defined in [RFC1305] Appendix C as a 96-bit field.

The **Authenticator** field defined by the NTP Authentication Extensions is a 160-bit field. When using this extension, the total NTP message length is 68 bytes. Windows clients and domain controllers rely on the NTP message length to detect the use of this extension. For more information about the **Authenticator** field, see sections 2.2.1 and 2.2.2.

The key identifier in the **Key Identifier** subfield of the **Authenticator** field identifies the cryptographic key used to generate the crypto-checksum. The association of key identifier and cryptographic key is mutually established between the client and server as a prerequisite, as explained in section 1.5.2. The algorithm to compute the crypto-checksum is specified in sections 3.1.5 and 3.2.5.

2.2.1 Client NTP Request

The subfields of the **Authenticator** field of the Client NTP Request message include the following.

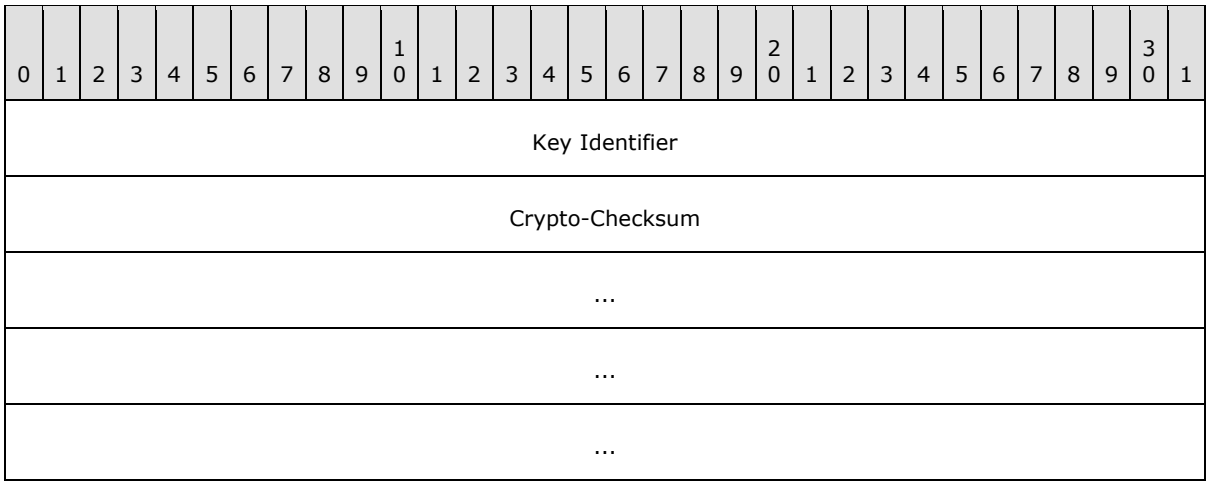


Key Identifier (4 bytes): A 32-bit unsigned integer in little-endian byte order (least significant byte first). This field identifies the cryptographic key used to generate the crypto-checksum. The least significant 31 bits are the least significant 31 bits of a RID for a trusted account, as described in section 1.5.2. The most significant bit of the key identifier is a 1-bit key selector that identifies the cryptographic key to use in the pair of keys associated with the account. For more information on the semantics of this key selector, see sections 3.1.5 and 3.1.6.

Crypto-Checksum (16 bytes): A 128-bit crypto-checksum that the encryption procedure computes. Recipients of a Client NTP Request message MUST ignore this subfield.<2>

2.2.2 Server NTP Response

The subfields of the **Authenticator** field of the Server NTP Response message include the following.



Key Identifier (4 bytes): A 32-bit unsigned integer in little-endian byte order (least significant byte first). It identifies the cryptographic key used to generate the crypto-checksum, as specified in section 2.2.1. Recipients of a Server NTP Response message MUST ignore this subfield.<3>

Crypto-Checksum (16 bytes): A 128-bit crypto-checksum computed by an encryption procedure. For more information, see section 3.2.5.

3 Protocol Details

The NTP Authentication Extensions operate between a client and a server during authenticated time synchronization.

3.1 Client Details

3.1.1 Abstract Data Model

Authenticated: This Boolean variable tracks whether the last incoming response was authenticated successfully.

Trusted Domain: This variable specifies a null-terminated Unicode string that represents the name of a trusted domain in which a secure connection can be established between the client and server.

Key Selector: This variable specifies the index of the crypto-checksums that the client is currently requesting for use in authenticating responses. The crypto-checksums are computed by the Netlogon message protection method using the cryptographic keys.

RID: This variable specifies the RID that uniquely identifies a trusted account whose passwords are used for establishing the secure connection in the trusted domain.

3.1.2 Timers

No new timers are required beyond those in NTP Version 3, as specified in [RFC1305] section 3.2.3. The client polling timer, as specified in [RFC1305] section 3.2.3, is used to control the polling interval between transmitted Client NTP Request messages. The polling interval for this timer varies in a dynamic range between a minimum polling interval and a maximum polling interval, as specified in [RFC1305] section 3.4.2.<4>

3.1.3 Initialization

The client initializes the *Trusted Domain* variable by calling the NetrLogonGetTimeServiceParentDomain method (as specified in [MS-NRPC] section **3.5.4.6.4**) with the following input parameters:

- *ServerName* MUST be set to NULL.

Upon successful completion, the output parameter *DomainName* will contain the name of a trusted domain in which a secure connection can be established between the client and server.

The client initializes the *RID* variable by calling the NetrLogonGetTrustRid method (as specified in [MS-NRPC] section **3.5.4.6.1**) with the following input parameters:

- *ServerName* MUST be set to NULL.
- *DomainName* MUST be set to the value of the *Trusted Domain* variable.

Upon successful completion, the output parameter *Rid* will contain the RID that uniquely identifies a trusted account.

The client initializes the *Authenticated* variable to true, initializes the *Key Selector* variable to zero, and starts the client polling timer.

3.1.4 Higher-Layer Triggered Events

No higher-layer triggered events are used.

When an end user configures NTP time synchronization without authentication, the NTP Authentication Extensions are not used. When an end user configures authenticated NTP time synchronization, the NTP Authentication Extensions are used. The configuration is the responsibility of the implementers.

When the W32TimeSync method specified in [MS-W32T] is invoked, the client polling timer specified in section 3.1.2 expires immediately.

3.1.5 Message Processing Events and Sequencing Rules

The following diagram illustrates the client logic for processing a Server NTP Response message received in response to a Client NTP Request message requesting authentication.

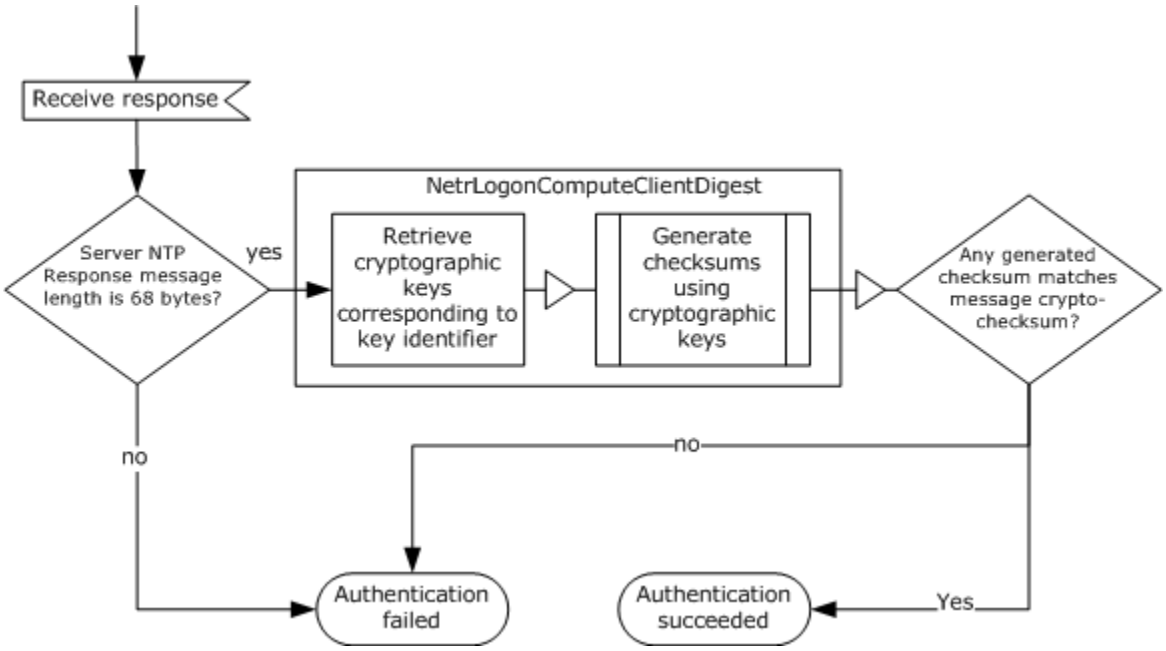


Figure 1: Client logic for processing Server NTP Response message

<5>

The client ignores the **Key Identifier** subfield of the Server NTP Response message.

The client uses the NetrLogonComputeClientDigest method (as specified in [MS-NRPC] section 3.5.4.7.3) to compute crypto-checksums for the first 48 bytes of the Server NTP Response message, with the following input parameters:

- *ServerName* MUST be set to NULL.
- *DomainName* MUST be set to the value of the *Trusted Domain* variable.
- *Message* MUST refer to the first 48 bytes of the Server NTP Response message.
- *MessageSize* MUST be set to 48.

The NetrLogonComputeClientDigest method computes two crypto-checksums using the pair of passwords associated with the trusted account.

The client compares each computed crypto-checksum with the **Crypto-Checksum** subfield in the Server NTP Response message. If the **Crypto-Checksum** subfield matches any of the computed crypto-checksums, the authentication succeeds. Otherwise, the authentication fails. A client SHOULD compare all computed crypto-checksums before determining that the authentication has failed; however, it SHOULD NOT continue to compare crypto-checksums after it has determined that at least one of its computed crypto-checksums matches the **Crypto-Checksum** subfield.

If authentication succeeds, the client continues processing the response to synchronize time the same way it occurs in the base NTP protocol, as specified in [RFC1305] section 3.4.3.

If authentication fails, the response MUST be ignored, and the client MUST NOT perform time synchronization using the response.

The following state diagram illustrates updates to the client's state variables based on success or failure of message authentication.

Notes on the following state diagram: In authenticated NTP, all state transitions are triggered by timer expiry. On expiration of the client polling timer, an authenticated NTP client attempts an authenticated NTP exchange with the NTP server. Based on the success or failure of that attempt, it updates state variables and transitions to the next state. The labels on the following arcs indicate the trigger of authentication success or failure that causes transition to the next state. For each trigger of authentication success or failure, there is an implicit trigger of "Timer Expiry" because it is the expiration of the polling timer that causes an authentication attempt. The "Timer Expiry" label has been omitted from the following arcs for clarity. Also note that the state variable assignments in the state boxes are carried out upon entry into the state, not on exit.

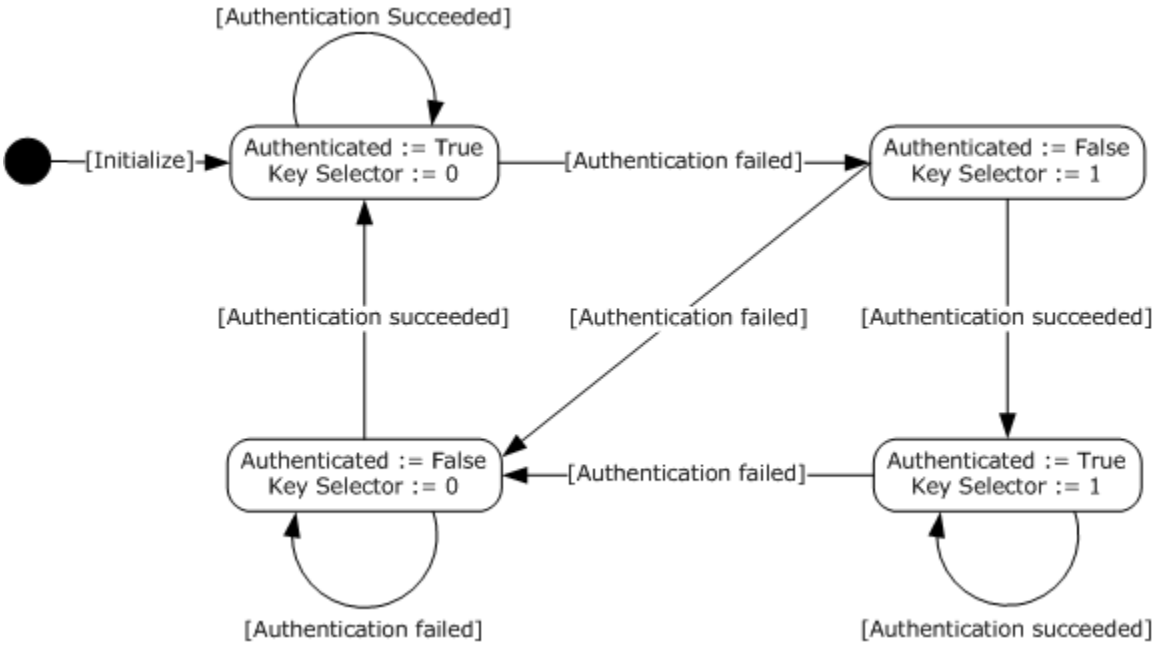


Figure 2: Updates to the client state variables

3.1.6 Timer Events

When the client polling timer expires, the client prepares a Client NTP Request message header in the same way it occurs in the base NTP protocol, as specified in [RFC1305] section 3.4.2. However, to request authentication, the Client NTP Request message length is 68 bytes. The client sets the **Authenticator** field of the Client NTP Request message, as specified in section 2.2.1, writing the least significant 31 bits of the *RID* value into the least significant 31 bits of the **Key Identifier** subfield of the authenticator, and then writing the *Key Selector* value into the most significant bit of the **Key Identifier** subfield.<6>

The client sends the Client NTP Request message to the server as it does in the base NTP protocol, as specified in [RFC1305] section 3.4.2.

3.1.7 Other Local Events

No additional events are used.

3.2 Server Details

3.2.1 Abstract Data Model

Accounts: This variable represents a list of trusted accounts that are indexed by the RID in a Windows domain. Each RID is a key identifier that identifies a trusted account. Each trusted account is associated with a pair of cryptographic keys that are the NTOWFv1 (as specified in [MS-NLMP] section 3.3.1) of the old and new passwords for the account.

3.2.2 Timers

No new timers are required beyond those in the base NTP protocol, as specified in [RFC1305] section 3.2.3.

3.2.3 Initialization

The server initializes the Authentication Keys list with all the associations of key identifier and cryptographic key. The server sets the Key Identifier value to the key identifier, and it sets the Cryptographic Key value to the cryptographic key.

The association of key identifier and cryptographic keys is mutually established between the client and the server, as specified in section 1.5.2.

At initialization time, the server begins listening for messages, as specified in [RFC1305] section 3.4.3.

3.2.4 Higher-Layer Triggered Events

In the default configuration of a domain-joined Windows NTP client, the use of the **W32TimeSync** method (as described in [MS-W32T] section 3.1.4.1) results in the higher-layer triggered use of NTP with authentication extensions (also noted in [MS-W32T] section 1.6). This is because the **W32TimeSync** method causes an NTP client to immediately synchronize time with its time source, and in the default configuration of a domain-joined client, this action results in the [MS-SNTP] server receiving a client time sync request using NTP with authentication extensions.

3.2.5 Message Processing Events and Sequencing Rules

The following diagram illustrates the server logic for providing authentication when responding to a Client NTP Request message using this extension. Other NTP processing is as specified in [RFC1305] section 3.4.3.

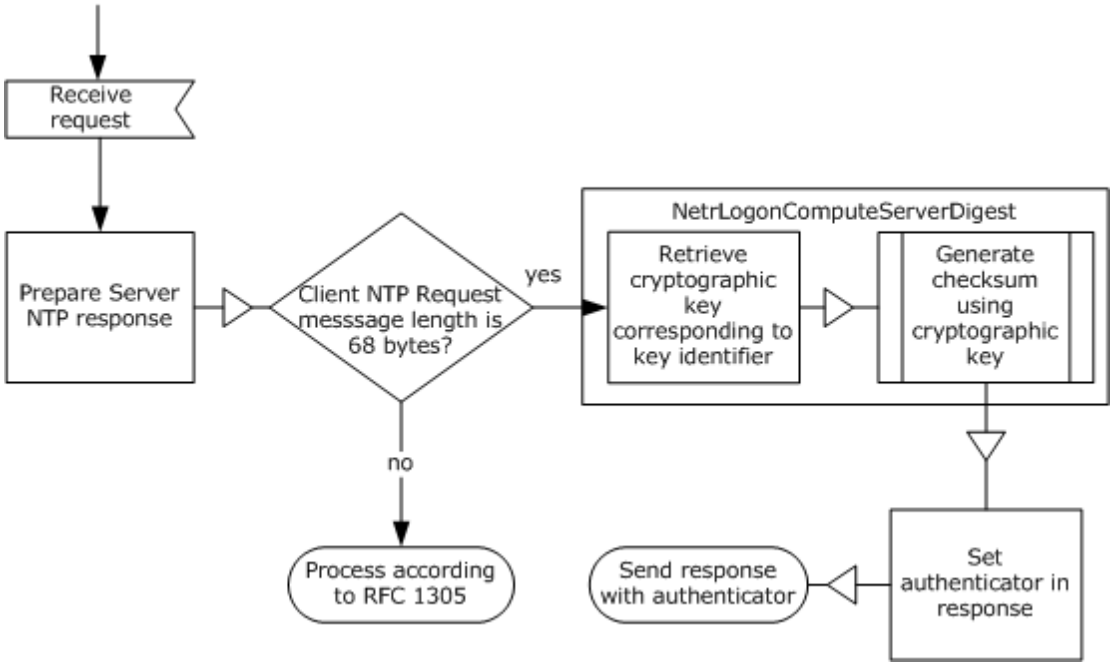


Figure 3: Authentication

When the server receives the Client NTP Request message from the client, the server examines the NTP message length. The **Authenticator** field that the NTP Authentication Extensions specify is present if the NTP message length is 68 bytes. In this case, the server MUST respond with an authenticated NTP response. Any other message length is processed as specified in [RFC1305] section 3.4.3.<7>

The server prepares the Server NTP Response message in the same way as for the base protocol, as specified in [RFC1305] section 3.4.3. Afterward, the server follows the server-encryption procedure.

The server ignores the **Crypto-Checksum** subfield of the Client NTP Request message.

The server retrieves the RID from the least significant 31 bits of the **Key Identifier** subfield of the **Authenticator** field of the Client NTP Request message. The server uses the NetrLogonComputeServerDigest method (as specified in [MS-NRPC] section **3.5.4.6.2**) to compute crypto-checksums with the following input parameters:

- *ServerName* MUST be set to NULL.
- *Rid* MUST be set to the RID value retrieved from the Client NTP Request message.
- *Message* MUST refer to the first 48 bytes of the Server NTP Response message.
- *MessageSize* MUST be set to 48.

The NetrlogonComputeServerDigest method looks up the trusted account identified by the RID in the Accounts table and computes two crypto-checksums using the pair of passwords associated with the trusted account.

If the Netlogon Remote Protocol method fails, the server SHOULD<8> fail the authentication and ignore the request without responding.

The server retrieves a 1-bit key selector from the most significant bit of the **Key Identifier** subfield of the **Authenticator** field in the Client NTP Request message. If the 1-bit key selector is set to 0, the server selects the crypto-checksum that was computed using the new password. If the 1-bit key selector is set to 1, the server selects the crypto-checksum that was computed using the old password.<9>

The server sets the **Authenticator** field of the Server NTP Response message as specified in section 2.2.2, writing the computed crypto-checksum into the **Crypto-Checksum** subfield of the **Authenticator**.

3.2.6 Timer Events

No timer events are used.

3.2.7 Other Local Events

No additional events are used.

4 Protocol Examples

The following example shows a successful authenticated time synchronization sequence with the NTP Authentication Extensions between Client C and Server S.

Note that the following packet diagrams illustrate the entire NTP packet, not just the **Authenticator** field in the NTP Authentication Extensions. The NTP data format is defined in the base NTP protocol, as specified in [RFC1305] Appendix A. The **Authenticator** field related to the NTP Authentication Extensions is specified in section 2.2. The packet diagrams use fictitious values for the **Key Identifier** and **Crypto-Checksum** fields.

The following example is also Windows-specific and shows how a Windows client and a Windows time server interact.

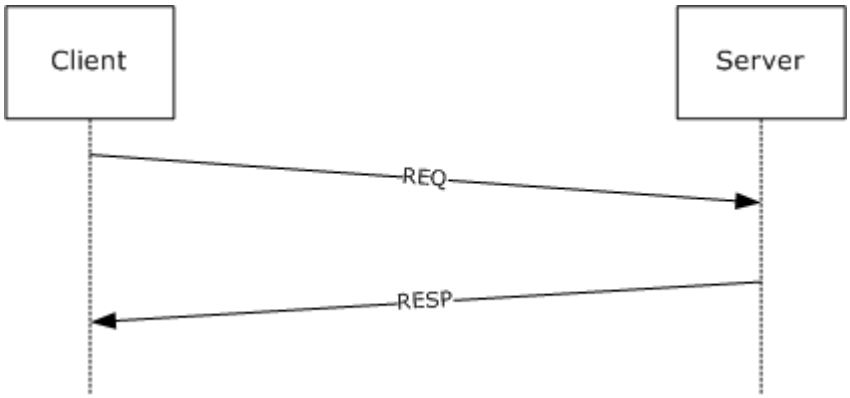


Figure 4: Message sequence

1. C waits for the client polling timer to expire.

2. C composes a Client NTP Request message. It sets the **Key Identifier** subfield of the extension part with the RID and the *Key Selector* value, and sets the **Crypto-Checksum** subfield to 0. C sends the Client NTP Request message to S.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
LI		VN			Mode=3			Stratum								Poll Interval								Precision							
Root Delay																															
Root Dispersion																															
Reference Clock Identifier																															
Reference Timestamp																															
...																															
Original Timestamp																															
...																															
Receive Timestamp																															
...																															
Transmit Timestamp																															
...																															
0x00 0x35 0x7B 0x9D																															
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00																															

3. After receiving the request, S verifies that the received message size is 68 bytes. If it is not, S either drops the request (if the message size does not equal 48 bytes) or treats it as an unauthenticated request (if the message size is 48 bytes). Assuming that the received message size is 68 bytes, S extracts the RID from the received message. S uses it to call the NetrLogonComputeServerDigest method (as specified in [MS-NRPC] section **3.5.4.6.2**) to compute the crypto-checksums and select the crypto-checksum based on the most significant bit of the **Key Identifier** subfield from the received message, as specified in section 3.2.5. S then sends a response to the client, setting the **Key Identifier** field to 0 and the **Crypto-Checksum** field to the computed crypto-checksum.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
LI		VN			Mode=4			Stratum								Poll Interval								Precision							
Root Delay																															
Root Dispersion																															
Reference Clock Identifier																															
Reference Timestamp																															
...																															
Original Timestamp																															
...																															
Receive Timestamp																															
...																															
Transmit Timestamp																															
...																															
0x00 0x00 0x00 0x00																															
0x5E 0xBD 0xA9 0x0E 0xB2 0x35 0x54 0x9A 0xB2 0xA8 0xB3 0x4F 0x50 0x1D 0x62 0x20																															

4. C receives the response and, after it verifies that the message size is 68 bytes, computes the crypto-checksums, as specified in section 3.1.5. C finds that one of the calculated crypto-checksums matches the crypto-checksum in the response. The response is, therefore, authenticated successfully, and C synchronizes its time.

5 Security

5.1 Security Considerations for Implementers

The cryptographic hash is calculated using the Netlogon Remote Protocol message protection methods. The methods use the MD5 algorithm, which is considered vulnerable to brute-force collision attacks. For more imformation on MD5 collisions, see [MD5Collision].

NTP Authentication Extensions provide secure Server NTP Response messages, but do not prevent invalid ones. For example, when the NTP server's own time is misconfigured, a Server NTP Response message is sent with an invalid time to the client. Such an invalid Server NTP Response message could lead to the client being synchronized to an invalid time. The client SHOULD check the time difference between the client and the server, and SHOULD synchronize time only if the time difference is within some reasonable range.<10>

5.2 Index of Security Parameters

Security parameter	Section
Keys and key identifier	1.5.2
Client NTP Request message	2.2.1
Server NTP Response message	2.2.2

6 Appendix A: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows 2000
- Windows XP
- Windows Server 2003
- Windows Vista
- Windows Server 2008

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that Windows does not follow the prescription.

<1> Section 1.3.2: Windows 2000 implements time synchronization based on SNTP, as specified in [RFC2030]. Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008 implement time synchronization based on NTP, as specified in [RFC1305].

<2> Section 2.2.1: Windows clients set this field to 0, and Windows servers ignore this field.

<3> Section 2.2.2: In Windows Server 2003, Windows domain controllers set this field to 0. In Windows 2000 and Windows Server 2008, Windows domain controllers set this field to the value of the Key Identifier subfield of the Client NTP Request message.

<4> Section 3.1.2: In Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008, the minimum polling interval and the maximum polling interval are configurable and vary between domain roles (member machine versus domain controller). By default, for a member machine acting as an NTP client, the minimum polling interval is 10 and the maximum polling interval is 15; for a domain controller acting as an NTP client, the minimum polling interval is 6 and the maximum polling interval is 10. These interval values are expressed (as specified in [RFC1305] section 3.2.7) in units of seconds and are exponents to a power of two; thus, the default minimum polling interval for a domain controller is 2 ^ 6 = 64 seconds, and the default maximum polling interval is 2 ^ 10 = 1,024 seconds.

Constants specifying the minimum (NTP.MINPOLL) and maximum (NTP.MAXPOLL) values permissible for a client's polling interval are as specified in [RFC1305] section 3.2.7. The Windows implementation defines different constants for the minimum and maximum permissible values. These constants are used to validate any values specified in configuration for the minimum polling interval and maximum polling interval. The following table shows the definitions of maximum (NTP.MAXPOLL) and minimum (NTP.MINPOLL) permissible values for a client's maximum and minimum polling intervals for different Windows versions.

Windows version	NTP.MAXPOLL: Domain controllers	NTP.MAXPOLL: Member machines	NTP.MINPOLL: All
Windows Server 2003	15	17	4
Windows XP	10	10	4

Windows version	NTP.MAXPOLL: Domain controllers	NTP.MAXPOLL: Member machines	NTP.MINPOLL: All
Windows Vista	15	17	4
Windows Server 2008	15	17	4

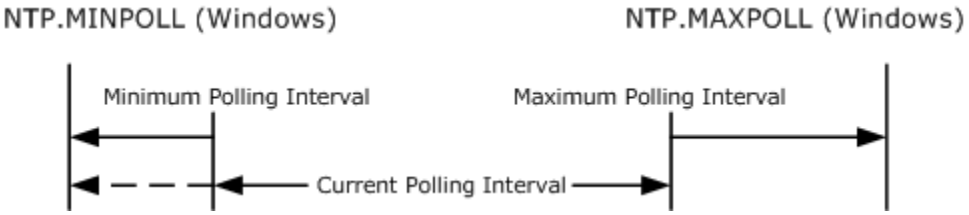


Figure 5: Polling intervals

In Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008, the **Poll interval** (as specified in [RFC1305] Appendix A) is initialized to NTP.MINPOLL. If the client continuously receives valid responses, the **Poll interval** is incremented from NTP.MINPOLL to no more than NTP.MAXPOLL. If the client fails to receive a valid response after three consecutive attempts, the **Poll interval** is decremented. If the client continues to fail to receive valid responses, the **Poll interval** is decremented further below the minimum polling interval, but never falls below the value defined for NTP.MINPOLL by Windows.

After eight consecutive failures to receive a valid response, the client pauses its synchronization attempts for a "back-off" interval (15 minutes), after which it returns to its initial **Poll interval**. The back-off interval is doubled for each subsequent occurrence of eight consecutive failures. This doubling occurs no more than six times for a maximum back-off interval of no more than 960 minutes.

Because of a bug in Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008, the client always incorrectly sets the **Poll interval** field of the first Client NTP Request message to the value defined for NTP.MAXPOLL by Windows.

Windows 2000 SNTP clients do not implement a true minimum or maximum polling interval. Instead, Windows 2000 clients initially poll by default every 45 minutes (the **Poll interval** value in the SNTP message is set to 11 for this phase). After three successful poll operations, Windows 2000 clients jump to polling every 8 hours (the **Poll interval** value is 14 for this phase). After every unsuccessful poll attempt, the interval reverts to 45 minutes.

<5> Section 3.1.5: In Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008, the NetrLogonComputeClientDigest method, as specified in [MS-NRPC] section 3.5.4.7.3, generates only two crypto-checksums for the current and previous password.

<6> Section 3.1.6: Windows 2000 clients do not use the most significant bit of the **Key Identifier** subfield, and always set the most significant bit to 0. In Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008, the client sets the **Key Identifier** subfield, as specified in section 2.2.1. The most significant bit is initialized to the value of the *Key Selector* abstract variable.

The client sets the **Crypto-Checksum** subfield, as specified in section 2.2.1.

In Windows 2000, the client always sets the **Mode** field of its Client NTP Request messages to 0x3 ("Client"). In Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008, a client

that is also a time source sets the **Mode** field of its Client NTP Request messages to 0x1 ("Symmetric Active"), while a client that is not a time source sets the **Mode** field in its Client NTP Request messages to 0x3 ("Client"). By default, a client running on a domain controller is also a time source. The syntax and semantics for the **Mode** field of the Client NTP Request message are specified in [RFC1305] Appendix A.

<7> Section 3.2.5: Windows 2000 Server ignores the Client NTP Request message when the NTP message length is less than or equal to 48 bytes. Windows Server 2003 and Windows Server 2008 ignore the Client NTP Request message when the NTP message length is neither 68 bytes nor 48 bytes.

<8> Section 3.2.5: Windows NTP servers in Windows 2000, Windows XP, and Windows Server 2003 do not honor the above "SHOULD". Instead, they respond to the request. In Windows 2000, the server responds with a Server NTP Response message without an **Authenticator** field if authentication fails. In Windows XP and Windows Server 2003, the server responds with a Server NTP Response message that includes an **Authenticator** field in which the **Crypto-Checksum** subfield is set to zero.

<9> Section 3.2.5: In the situation where the machine account has only a current password (that is, an old password does not yet exist) and a client requests a digest computed using the old password, Windows computes the digest using the current password. Windows 2000 is a special case in that it returns an unauthenticated response when an old password does not exist.

<10> Section 5.1: The client accepts any Server NTP Response message regardless of the time difference in authenticated NTP time synchronization inside a Windows domain.

7 Index

T

V

W