Extending the Network Time Security Protocol for Secure Communication between Time Server and Key Establishment Server

Martin Langer, Kai Heine, Rainer Bermbach
Ostfalia University of Applied Sciences
Wolfenbüttel, Germany
{mart.langer, ka.heine, r.bermbach}@ostfalia.de

Dieter Sibold
Physikalisch-Technische Bundesanstalt (PTB)
Braunschweig, Germany
dieter.sibold@ptb.de

***Abstract*—This work describes a concept for extending the Network Time Security (NTS) protocol to enable implementation-independent communication between the NTS key establishment (NTS-KE) server and the connected time server(s). It fills a specification gap left by RFC 8915 for securing the Network Time Protocol (NTP) and enables the centralized and public deployment of an NTS key management server that can support both secured NTP and secured PTP.**

Keywords—NTS; NTPv4; Network Protocol; Security

I. INTRODUCTION

Time information is essential for the reliable functioning of many computer systems. This information is needed to ensure interoperability between machines, to keep databases consistent or to utilize security mechanisms (e.g., certificate verification). Therefore, most devices use a time protocol to fetch time data from a public server or to synchronize their clocks within a network. A well-known and widely used representative is the Network Time Protocol (NTP) [1], which is one of the oldest packet-based Internet protocols that achieves synchronization accuracies in the lower millisecond range. In areas with higher requirements, the Precision Time Protocol (PTP) [2] is used instead, which enables accuracies in the nanosecond range. However, the security in NTP and PTP was neglected for a long time and offered a large potential for attack. In October 2020, an effective protection mechanism for NTP was made available in the form of the Network Time Security (NTS) protocol [3], which has since been integrated into many NTP implementations such as Chrony [4], NTPsec [5], Cloudflare NTS [6], and NTP-O [7]. Developments are also in progress on the PTP side to make use of NTS protection here as well [8].

The NTS standard allows the separation of the NTS-KE server (key management server) from the actual time server. This separation can take place both logically on a single machine and physically on different machines. It reduces the load on the time server because the establishment of secure TLS connections for the initial key distribution of the requesting clients is outsourced. When running more than one NTP server, further advantages become apparent. Since the NTS-KE server can manage multiple time servers as well as support server negotiation, this also enables load balancing mechanisms. In addition, a client that authenticates such an NTS-KE server has potential access to multiple NTS-secured time servers.

However, the NTS standard did not specify the communication between the NTS-KE server and the time servers (see Figure 1). As a result, all NTP services currently solve it implementation-dependently. Interoperability with a public and common NTS-KE server is thus not possible, since a defined communication would have to be specified for this. Another problem is the construction of the necessary cookies in NTS, which is not normatively described either and also prevents interoperability with a shared NTS-KE server at this level.

This paper proposes an extension to the NTS protocol that defines the communication between the NTS-KE server and the NTP time server along with describing possible solutions to the cookie problem. The communication relies on a new NTS subprotocol, which has also been described for the NTS-secured PTP connection [8].

In the further course of the paper, Chapter II provides some basics about NTS and discusses related work. Chapter III then describes the concept of communication between NTS-KE server and NTP time server, including communication flow, messages and cookie handling. The following chapter VI gives an overview of the main security features and finally chapter V concludes the content of this paper.

II. PRELIMINARIES AND RELATED WORK

This chapter provides a brief overview of the functions and characteristics of the Network Time Security protocol. It covers both the part specified in RFC 8915 for NTP and the part that is currently under development for PTP.

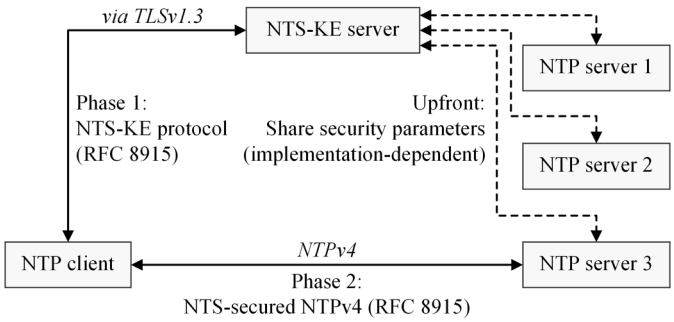

Fig. 1. Communication structure in RFC 8915

978-1-6654-3935-0/21/$31.00 ©2021 IEEE

A. NTS for NTP

The Network Time Security protocol is a security upgrade for time protocols that is currently standardized in RFC 8915 for NTP. It replaces both the pre-shared-key method defined in the NTPv4 standard [1], which is rarely used due to poor scaling, and the Autokey method [9], which is insecure according to analysis in [10]. NTS provides authenticity as well as integrity of timing messages, and offers protection against attacks on timing protocols defined in RFC 7384 [11], such as spoofing, replay, and packet tampering. By design, NTS for NTP is divided into two sub-protocols. The first is the NTS key establishment (NTS-KE) protocol, which defines the communication between NTS-KE server (equivalent to a key management server) and the NTP clients. For this, TLSv1.3 is used as the framework protocol to enable secure NTS message exchange over it. These messages allow the exchange of AEAD algorithms, server addresses, keys and cookies. The cookies are encrypted and contain keys and parameters as well, to which only the NTS-KE server and the NTP server have access. In the second protocol - the time protocol itself - these parameters and keys are now used to secure the communication between the NTP client and the NTP server. NTS defines NTPv4 extension fields for this purpose, in order to carry the security parameters and cookies along with integrity protection. The cookies allow the NTP server to check and secure the NTP messages and enable the stateless operation. If there are multiple errors or packet loss in NTP, the first TLS-based phase is executed again.

B. NTS for PTP

The NTS for PTP (NTS4PTP) protocol, which is currently being developed by us and submitted as an IETF draft [8], is intended to extend NTS to include PTP in the future. PTP messages are secured by the AUTHENTICATION TLV defined in the PTPv2.1 standard [2], so an NTS-based solution at the time protocol level is not necessary. However, the distribution of security parameters and keys is managed by the NTS-KE server, so that new NTS messages have been defined for this purpose. A characteristic of that draft is the extension of NTS by a further sub-protocol. In this, the NTS key management is extended by the NTS time server registration (NTS-TSR) protocol, which defines the communication between the NTS-KE server and the PTP time server for PTP unicast mode.

III. CONCEPT

The basic idea is to extend the NTS-TSR protocol designed in NTS4PTP to include communication between NTS-KE servers and the NTP time servers (see Figure 2). NTS4PTP defines so-called registration messages in NTS-TSR to transfer the payload data. Therefore, the use of these message types is also useful for NTP, which of course transports different information than in PTP. The next section describes the communication flow of the NTS-TSR protocol for NTPv4. This is followed by information about the message structure and the handling of the cookies.

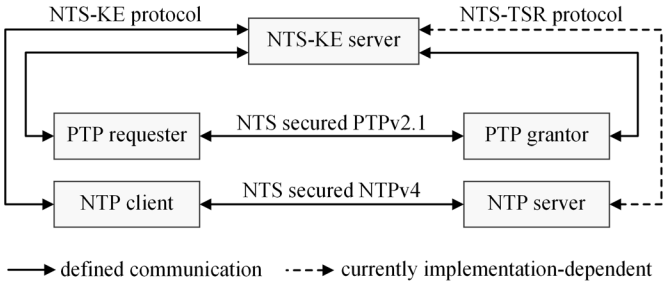

Fig. 2. Communication interfaces of NTS for NTP and PTP

A. Communication Flow

The basic communication flow is very similar to the procedure defined in NTS4PTP. Here too, the NTP server must first register with the NTS-KE server so that the NTP client can retrieve the necessary security parameters for the desired time server.

1) Phase 1: NTS-TSR Protocol

The communication therefore starts with the NTP server, which first establishes a TLSv1.3 connection to the NTS-KE server (see Figure 3). Via the Application-Layer Protocol Negotiation (ALPN) extension in TLS [12], the NTP server signals that it wants to speak the NTS-TSR protocol. Then, the communication partners authenticate each other based on their certificates. Furthermore, the NTS-KE server checks whether the NTP time server is authorized for further communication (see chapter IV). If this is the case, the TLS channel is established and the NTS-TSR protocol is executed.

In the next step, the NTP server sends an NTP Registration Request message to the NTS-KE server. This message is protected and encrypted by the TLS channel and essentially contains the parameters that the NTS-KE server needs to construct the NTS cookies. After receiving the message, the NTS-KE server first checks if the NTP server is already registered and if the parameters are supported. If this is an initial

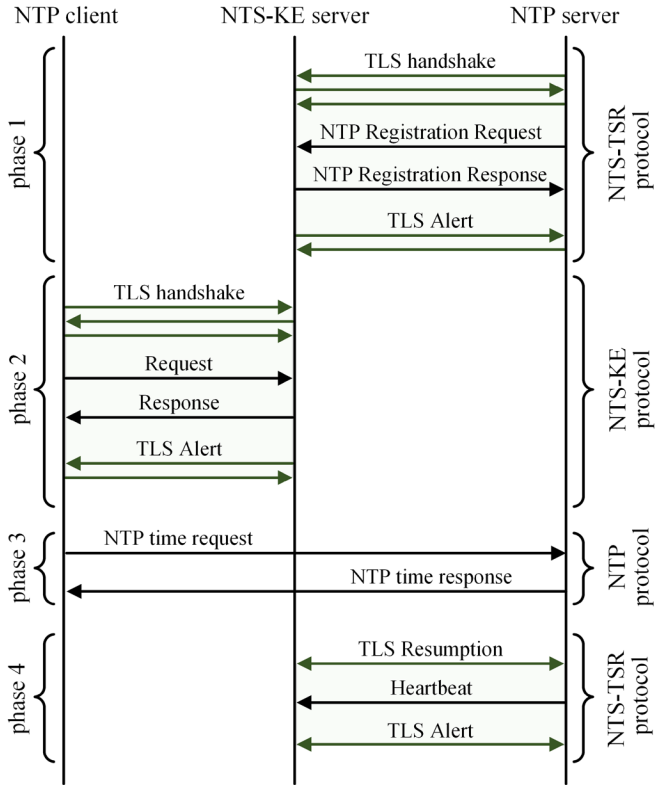

Fig. 3. Sequence of the NTS time server registration protocol for NTPv4

registration, the NTS-KE server accepts the time server and stores the received information. If the NTP server is already registered, the NTS-KE server updates some security parameters including key to ensure key freshness. After that, the NTS-KE server generates an NTP Registration Response message, which is sent back to the NTP server over the TLS channel and contains only a timeout parameter for heartbeat messages. Any keying material can be derived independently by both communication partners from the established TLS channel using the TLS keying material exporter function [13, ch. 7.5]. The NTS-KE server requires one symmetric key, which is needed for cookie encryption and is called Master Key A (MK_A). The NTP server, on the other hand, generates two symmetric keys. One is MK_A, which is used to check cookies generated by the NTS-KE server, and the other is Master Key B (MK_B), which is used to secure and check cookies generated by the NTP server itself (see also chapter 4). Immediately after sending the response message, both sides properly close the connection by sending a TLS Alert message containing a *close_notify* signal. This completes the first phase and any number of NTP clients are now able to request a secure connection to the NTP server.

2) Phase 2: NTS-KE Protocol

In the second phase, an NTP client now connects to the NTS-KE server and also starts establishing a TLS channel. In this process, the NTP client checks the authenticity of the NTS-KE server. Mutual verification is not necessary at this point, as a non-authenticated client does not jeopardize the infrastructure and security. The next step starts the execution of the NTS-KE protocol as described in RFC 8915. For this, the NTP client provides its supported crypto algorithms via the NTS request and asks for NTS cookies for the desired NTP server. If the NTP server is not known, the client can also be assigned to another time server by the NTS-KE server. If the NTS request is valid and the corresponding NTP time server is available, the NTS-KE server generates up to eight cookies and encrypts them with MK_A. In addition to the negotiated AEAD algorithm, these cookies contain a client-to-server (C2S) and a server-to-client (S2C) key that both communication partners can derive independently from the TLS connection using the same scheme. These keys are regenerated for each NTS request and are used to secure the NTP packets in the next phase. The NTS-KE server now sends the NTS response back to the NTP client, which contains the negotiated AEAD algorithm and the address of the time server in addition to the cookies. The NTP client and NTS-KE server close the TLS channel and thus end the second phase.

3) Phase 3: NTP Protocol

At the time protocol level, the client now creates an NTP request and, in addition to a unique identifier (UID) as replay protection, also embeds one of the eight cookies in the NTP message in the form of an NTP extension field. The NTP client then secures the NTP message with the C2S key and sends the packet to the NTP time server. The latter extracts the contained cookie and identifies that it has been encrypted with the MK_A based on the specified key ID. Next, NTP server decrypts that cookie, extracts the C2S/S2C keys it contains as well as the AEAD algorithm to be applied, and uses them to verify the NTP packet. If it is valid, the NTP server constructs a new cookie encrypted with MK_B, embeds it in an NTP response along with the unmodified UID, secures it with the S2C key, and sends it back to the client. After receiving and verifying the NTP message, the client applies the included time information and stores the new cookie. The communication period between the NTP client and the NTP server continues until the NTP client desires a key replacement or until all the held cookies have been consumed due to message loss. If this happens, the NTP client executes the NTS-KE protocol again.

4) Phase 4: Re-Registration and Heartbeat Messages

On the NTP server side, key freshness is based on re-registration. If the NTP server wants a master key rotation (e.g., daily), it connects to the NTS-KE server and executes the NTS-TSR protocol again. In addition to the MK_A and MK_B keys, all other security parameters can also be updated. Since the expired master keys may still be used for a certain period of time by the NTP server to check older cookies, there is also no interruption of active NTP communication.

To ensure that dead NTP servers do not remain listed in the NTS-KE server forever, a heartbeat message is used. An NTP server must send this to the NTS-KE server before the specified timeout expires. If it does so, the timeout resets and the NTP server remains listed. If the timeout expires, the NTS-KE server removes the NTP server from the list of available time servers and a new registration is required. The simplest variant to send the heartbeat messages would be again over the TLS channel. This makes the definition of further security measures like replay protection, authenticity or even integrity protection unnecessary. To minimize the overhead of one of the TLS handshakes, TLS resumption should be used for this purpose. For a re-registration, on the other hand, a fresh TLS connection must be established in order to guarantee key freshness.

B. New NTS-TSR Messages for NTPv4

This section introduces the new NTS messages and describes the parameters they comprise. Just as in the NTS-KE protocol and the messages in the NTS-TSR protocol for PTP, the new messages consist of a set of NTS records. Those data structures, described in RFC 8915, are TLV-like constructs, each containing a specific payload. The ordering of these records is arbitrary with the exception of the End of Message (EoM) record, which must always be the last record.

1) NTP Registration Request

This message type is used for NTP time server registration and consists of 6 records in total (see Figure 4):

- **NTS Message Type** consists of an identifier plus a version number and is used to identify the NTS message. In addition to the NTS-TSR messages for NTP presented here, some are also defined for PTP [8].

NTS Message Type	NTP Registration Request || v1.0
NTPv4 Time Server Addresses	{Address 1 || Address 2 || ...}
Supported AEAD Algorithms	{AEAD...384 || AEAD...256}
Master Key AEAD Algorithm	AEAD_AES_SIV_CMAC_512
Master Key ID A	41253
End of Message	

NTS record record payload (example value)

Fig. 4. Structure of the NTP Registration Request message

- **NTSv4 Time Server Addresses** includes as payload one or more addresses under which the NTP server can be reached. This could be an IPv4, IPv6 or a Fully Qualified Domain Name (FQDN). The individual addresses are combined and concatenated in a TLV-like form. In addition to the addresses, such a TLV also includes the port number (see Figure 5). After registering the NTP server, an NTP client can specifically request a desired address type.
- **Supported AEAD Algorithms** holds the AEAD algorithms supported by the NTP server, which are suitable for the later protection of the NTP message. Later, this list allows the NTP client to negotiate the AEAD algorithm during the NTS-KE protocol.
- **Master Key AEAD Algorithm** containing the AEAD algorithm that will be used to encrypt the cookies.
- **Master Key ID A** carries an arbitrary identifier defined by the NTS server. This key ID is embedded in a plain text area of the otherwise encrypted cookies, which the NTP server can later use to identify the associated master key A.
- **End of Message** is an empty record that marks the end of the NTS message.

2) NTP Registration Response

The NTP Registration Response message is received by the client after a successful registration. It consists of three records (see Figure 6):

- **NTS Message Type** consists identifier and version number and is used to identify the NTS message.
- **Heartbeat Timeout** specifies a relative time in seconds and indicates the time by which an NTP time server must have sent a heartbeat message.
- **End of Message** is an empty record that marks the end of the NTS message.

In case of an error, the NTP Registration Response message contains an error record instead of the heartbeat timeout, with a corresponding error code (see Figure 7). This is the case, for example, if the NTS-KE server does not support the algorithms of the NTP server or cannot process the message.

Address Type	IPv4
Address Length	4 octets
Address Value	41.154.24.215
UDP Port	123
End of Message	

Fig. 5. Structure of a single address information block

NTS Message Type	NTP Registration Response || v1.0
Heartbeat Timeout	900s
End of Message	

Fig. 6. Structure of the NTP Registration Response message (success)

3) NTP Registration Revoke

An NTP server can send this message to the NTS-KE server if it wants to be removed from the list of available time servers before the heartbeat period expires. This might be the case in a situation of overload or maintenance work. The message contains only the records NTS Message Type and EoM (see Figure 8). The unambiguous assignment of this message to the listed NTP server at the NTS-KE server is achieved by the certificate of the sender.

4) Heartbeat

At the moment this message also contains only the NTS Message Type and an EoM (see Figure 9). However, this message is suitable for the transmission of additional information by a suitable record. For example, this could be status information of the NTP server in order to optimize the load management of the time servers connected to the NTS-KE. Without this information, simple load balancing via Round Robin would be possible.

C. Cookie Construction

Constructing NTS cookies for NTP is not straightforward. NTS defines a cookie format in RFC 8915, but it is not normative and has only been specified at a high level of abstraction. All existing NTS implementations follow the approach defined in NTS and use the attributes specified there, but have structural variations. In particular, the field lengths and byte order are different as well as the algorithm support. To solve this problem there are two possibilities.

A first variant is the use of cookie blueprints. This method allows the construction of different cookie formats, but is more complex. A blueprint consists of a construction plan that includes three substructures: general parameters, plaintext content, and encrypted content. The general parameters contain information such as the AEAD algorithm used to encrypt the cookie or the byte order. The plaintext contents include information such as the nonce, the master key ID or associated data. The encrypted contents include sensitive data such as the C2S and S2C keys and the negotiated AEAD algorithm between the NTP client and the NTP server. The individual components are specified in a blueprint in the form of TLV-like data blocks. One such block is a fixed data structure that specifies, in particular, the position, length, and type of a cookie field. However, for the implementation, a corresponding NTS record would have to be defined in the NTP Registration Request message.

NTS Message Type	NTP Registration Response || v1.0
Error	*association port not registered*
End of Message	

Fig. 7. Structure of the NTP Registration Response message (error)

NTS Message Type	NTP Registration Revoke || v1.0
End of Message	

Fig. 8. Structure of the NTP Registration Revoke message

NTS Message Type	Heartbeat || v1.0
Status	*server load: low*
End of Message	

Fig. 9. Structure of the Heartbeat message

The second and far better variant is the clean specification of an NTS cookie, which every NTS implementation must fulfill if it wants to interact with a public NTS-KE server. This is exclusively an implementation-specific problem that can also be easily addressed. Therefore, this variant should also be followed in a specification process and implemented as an RFC 8915 update.

IV. Security Considerations

The new NTS messages are generally secured via the TLSv1.3 channel. This protects against typical attacks such as spoofing, packet manipulation or even packet replay. For a TLS connection to work, this requires loose time synchronization of the NTS-KE server so that certificate checking works. The time must therefore be obtained via secured and redundant channels as well.

In addition to authentication, authorization is also an important factor in certificate checking. Thus, it must be ensured that only authorized time servers can join the NTS-KE server. This is important because otherwise attackers could register with the NTS-KE server with a certificate issued by LetsEncrypt, for example, and distribute false time information. The RootCA would therefore have to be an authority or otherwise trusted entity that only accepts verified time servers.

Since a public NTS-KE server is a prioritized attack target, mitigation measures must also be taken to mitigate denial-of-service attacks. Therefore, one option is to limit requests and ensure that the infrastructure is redundant. This means that multiple NTS-KE servers should be used to ensure better load balancing and resilience. However, Byzantine errors must be considered when synchronizing data between multiple NTS-KE servers.

The use of two master keys (MK_A and MK_B) is necessary in the NTS-TSR protocol because nonces are used in the encryption functions. A nonces is an arbitrary, often random number that may only be used once in cryptographic functions and could lead to a security breach if reused. Since NTS-KE servers and NTP servers independently generate the nonces, a collision cannot be ruled out. The use of two different master keys solves this conflict. It is the same procedure as for the C2S and S2C keys.

V. Conclusion

The NTS time server registration protocol is a useful extension for the current NTS specification. It solves the specification gap of the NTS specification in the communication between NTS-KE server and NTP time server and follows the structure defined in NTS4PTP draft. The combination of our solution with NTS4NTP (RFC 8915) and NTS4PTP provides a fully comprehensive protection for NTP and PTP based on NTS.

References

[1] D. L. Mills, U. Delaware, J. Martin, J. Burbank and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification," RFC 5905, doi 10.17487/rfc5905, 2010.

[2] IEEE Standards Association, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," IEEE Std 1588-2019, Jun. 2020.

[3] D. Franke, D. Sibold, K. Teichel, M. Dansarie and R. Sundblad, "Network Time Security for the Network Time Protocol," RFC 8915, doi 10.17487/rfc8915, Sep. 2020.

[4] M. Lichvar, "Chrony project," URL: https://github.com/mlichvar/chrony, 2021.

[5] E. S. Raymond, G. E. Miller, "NTPsec project," URL: https://github.com/ntpsec/ntpsec, 2021.

[6] W. Ladd, "cloudflare NTS project (cfnts)," URL: https://github.com/cloudflare/cfnts, 2021.

[7] M. Langer, et al., "NTP-O: first NTS capable NTP implementation," URL: https://gitlab:com/MLanger/ntp, 2019.

[8] M. Langer, R. Bermbach, "NTS4PTP - Key Management System for the Precision Time Protocol Based on the Network Time Security Protocol," Internet-Draft, draft-langerntp-nts-for-ptp-01, 2021.

[9] D. L. Mills, U. Delaware and Brian Haberman, "Network Time Protocol Version 4: Autokey Specification," RFC 5906, doi 10.17487/rfc5906, June 2010.

[10] S. Röttger, "Analysis of the NTP Autokey Procedures," Project Thesis, Technische Universität Braunschweig, Institute of Theoretical Computer Science, Braunschweig, 2012.

[11] T. Mizrahi, "Security Requirements of Time Protocols in Packet Switched Networks," RFC 7384, doi 10.17487/rfc7384, Oct. 2011.

[12] S. Friedl, A. Popov, A. Langley, S. Emile, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension," RFC 7301, doi:10:17487/RFC7301, 2014

[13] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, doi:10:17487/RFC8446, 2018.