ntp
Internet-Draft
Intended status: Experimental
Expires: 8 May 2026

D. Venhoek
Trifecta Tech Foundation
F. D. Vries
M. Schoolderman
Tweede golf B.V.
4 November 2025

NTS extensions for enabling pools
draft-venhoek-nts-pool-04

Abstract

The aim of this document is to describe a proof of concept system for NTS pools that are able to be used by clients without any knowledge beyond plain NTS. The work here focuses purely on creating an intermediate NTS Key Exchange server that can be configured with the addresses of multiple servers and distribute load between them. The parts of pool operation dealing with managing the list of servers are left out of scope for this work.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://pendulum-project.github.io/nts-pool-draft/draft-venhoek-nts-pool.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-venhoek-nts-pool/.

Source for this draft and an issue tracker can be found at https://github.com/pendulum-project/nts-pool-draft.

Status of This Memo

This Internet-Draft will expire on 8 May 2026.

Table of Contents

1. Introduction

NTS [RFC8915] provides authenticity and limited confidentiality for NTP [RFC5905]. However, the key exchange preceding the actual time exchange makes it hard to implement a pool for NTS supporting servers in a manner similar to the DNS resolution approach taken to provide the NTP Pool [Pool].

This document aims to provide extensions to the NTS Key Exchange sessions that allow for an implementation of a pool for NTS that:

* is usable without changes to the client,

* avoids constraining the time source’s cookie format,

* avoids time sources having potential access to all traffic.

2. Conventions and Definitions

Throughout the text, the terms client and server will refer to those roles in an NTS Key Exchange session as specified in [RFC8915]. Please note that this means that the pool itself operates in both roles: As a server towards users of the pool, and as a client towards the time sources.

Where further specificity of the role of a participant is needed, we will use the term user to indicate a user of a pool, the term pool to indicate the pool itself, and time source for the time servers that the pool delegates the actual providing of time to.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. General pool architecture

We propose a pool model where the pool provides an NTS Key Exchange service to the outside world. A major advantage of this model is that it avoids having to distribute certificates to all time sources. Contrary to [RFC8915], there is no direct TLS connection between the client and the selected time source.

In [RFC8915], cookies are generated based on key material that is extracted from this TLS connection. Our proposed model instead establishes two TLS connections: between the client and the pool, and between the pool and the time source. Because cookies need to be

generated using key material from the client, the pool extracts this key material and sends it to the time source. The time source uses this key material (rather than key material extracted from its connection with the pool) to generate cookies. This way, the pool can remain oblivious to the cookie format of the time source.

4. Communication between the pool and time sources

To facilitate communication between the pool and the time sources, 4 new NTS records are defined in Section 6. Together these records provide a way for the pool to provide key exchange services to clients on behalf of the time sources.

The Supported Next Protocol List (Section 6.2), Supported Algorithm List (Section 6.3) and List Server Names (Section 6.4)) records allow the pool to ask a time source which protocols and algorithms it supports, and which server names are used in the NTP server records it generates. This information can be requested by the pool at any time, and can be cached for short periods of time to improve efficiency.

Using knowledge of a time source’s supported protocols and algorithms, the pool can then handle client connections for that time source, using the clients indicated desires to choose a concrete next protocol and AEAD algorithm. The pool can then extract the keys from the TLS connection and use the Fixed Key record (Section 6.5) to request cookies for these keys from the time source. The response to a request containing a Fixed Key record will be the same as that for any regular NTS Key Exchange response, with the exception that the keys will be taken from the Fixed Key record instead of being derived from the TLS connection.

The list of server names provided by the time source can be used by the pool to honor requests by the client to not repeat a certain server. This allows more efficient retrieval of multiple sources from a pool.

As it is wasteful to setup a new TLS session between the pool and the time source for each of these interactions. To facilitate reuse of the TLS sessions, we further introduce the Keep Alive record (Section 6.1). This record allows the pool to indicate to the time source a desire to keep the session alive for more than a single request-response interaction.

4.1. Authenticating the pool to time sources

Allowing arbitrary clients to keep connections alive for more that a single request-response interaction could open up the server to denial of service due to resource exhaustion. To prevent this, a pool wishing to use the keep alive functionality MUST authenticate itself to the time source using an Authentication Token record (Section 6.7). Time sources MUST check that the content of the Authentication Token record matches the authentication string of a client that is on the list of requestors allowed to use the keep alive mechanism. By default, the list of requestors allowed to use the keep alive mechanism MUST be empty

Furthermore, time sources MAY choose to also restrict the Fixed Key, Supported Next Protocol List and Supported Algorithm List to authenticated clients. If this choice is made, it is suggested that the server treat these records as unrecognized critical records on unauthenticated client’s connections.

5. Communication between clients and the pool

A client requesting time from the pool can make a normal NTS Key Exchange request to the pool. In the response to the client the pool needs to tell which NTP server is to be used to get the time. This can be done through the already existing NTP Server Record. However, the pool needs to ensure it is present, and therefore MUST add such a record to the response unless one is already provided by the time source.

Clients that are aware they are talking to a pool may want to get multiple independent time sources from that pool. For this, they need to be able to tell the pool which time sources they already have, otherwise they might get a time source that they are already talking to. To achieve this, a client can use the NTP Server Deny record (Section 6.6) to indicate it would rather not receive a particular server. Clients MUST use the precise name given by the pool in a previous NTP Server record, otherwise the pool may not recognize which time source the client is referring to.

6. New NTS record types

6.1. Keep Alive

Record Type Number: To be assigned by IANA (draft implementations: 0x4000) Critical bit: 0

Indicates a desire to keep the TLS connection active for more than one message exchange. This can be used by a pool to reuse connections to a time source's NTS Key Exchange servers multiple times, reducing load on both the pool and time sources.

When a client sends this record the body MUST have size 0. A client MUST NOT use Keep Alive unless the request contains a record type allowing the use of Keep Alive. Within this specification, that is limited to the Supported Protocol List and Fixed Key Request records. A server SHOULD ignore any body for the Keep Alive record.

When supported by a server and allowed for the request in question, the server MAY include a Keep Alive record with a body of size 0 in the response and keep the TLS connection active after the response to handle further requests from the client. A client SHOULD ignore any body for the Keep Alive record. As keeping a connection active requires additional resources on the server, a server SHOULD NOT respond with a Keep Alive record to unauthenticated clients.

When included in the request or response, the client respectively server MAY, contrary to the requirements in [RFC8915], send another request or response. Any TLS "close_notify" SHALL be sent only after the last request or response respectively to use the connection.

Once a Keep Alive record has been sent by a client, or honored by a server, the TLS connection over which it was sent MUST NOT be used for key extraction. Doing so anyway can result in the reuse of keys and may result in loss of confidentiality or authenticity of the resulting NTP exchanges.

6.2. Supported Next Protocol List

Record Type Number: To be assigned by IANA (draft implementations: 0x4004) Critical bit: 1

This record can be used by a pool to query time sources about which next protocols they support.

When a client sends this record the body MUST have size 0. Clients MAY use Keep Alive in combination with this record. Contrary to [RFC8915], a request with this record SHOULD NOT include a "Next Protocol Negotiation", "AEAD Algorithm Negotiation" or "Fixed Key Request" record.

When receiving this record, servers MUST ignore any client body sent and MUST send in the response a Supported Next Protocol List record with as data a list of 16-bit integers, giving the protocol IDs the server supports. A server MAY treat this record as unknown for clients that are not authenticated as described in Section 4.1.

When included, the server MUST NOT negotiate a next protocol, AEAD algorithm, or keys for this request.

6.3. Supported Algorithm List

Record Type Number: To be assigned by IANA (draft implementations: 0x4001) Critical bit: 1

This record can be used by a pool to query time sources about which AEAD algorithms they support.

When a client sends this record the body MUST have size 0. Clients MAY use Keep Alive in combination with this record. Contrary to [RFC8915], a request with this record SHOULD NOT include a "Next Protocol Negotiation", "AEAD Algorithm Negotiation" or "Fixed Key Request" record.

When receiving this record, servers MUST ignore any client body sent and MUST send in the response a Supported Algorithm List record with as data a list of tuples of two 16-bit integers, the first giving an algorithm ID for the AEAD and the second giving the length of the key for that algorithm ID. A server MAY treat this record as unknown for clients that are not authenticated as described in Section 4.1.

When included, the server MUST NOT negotiate a next protocol, AEAD algorithm, or keys for this request.

We include the algorithm key size in the response so that a pool does not itself need knowledge of which AEAD algorithms exist, and what their key sizes are. Instead, it can use the provided key length when extracting keys from the TLS connection between end user and pool. This allows adoption of new AEAD algorithms without any changes to the pool software.

6.4. List Server Names

Record Type Number: To be assigned by IANA (draft implementations: 0x4005) Critical bit: 1

This record can be used by a pool to query time sources about which server names they use in NTP server records in their responses.

When a client sends this record the body MUST have size 0. Clients MAY use Keep Alive in combination with this record. Contrary to [RFC8915], a request with this record SHOULD NOT include a "Next Protocol Negotiation", "AEAD Algorithm Negotiation" or "Fixed Key Request" record.

Servers MUST NOT include this record in a response. When receiving this record, servers MUST ignore any body of this record sent by the client, and MUST send in the response one NTP server record for each server name the server may use responses to fixed key requests. If a server never sends a NTP server record in response to a fixed key request, it MAY opt to not provide one in response to this record.

When receiving this record, a server MUST NOT negotiate a next protocol, AEAD algorithm, or keys for this request. A server MAY treat this record as unknown for clients that are not authenticated as described in Section 4.1.

6.5. Fixed Key Request

Record Type Number: To be assigned by IANA (draft implementations: 0x4002) Critical Bit: 1

When a client is properly authenticated, the server SHOULD NOT perform Key Extraction but rather use the keys provided by the client in the extension field. In all other aspects, the response SHALL be the same as that from a regular key exchange session as specified in [RFC8915]. This allows a pool to do key negotiation on behalf of its users with the time source’s NTS Key Exchange servers, even though it terminates the TLS connection.

When used, the client MUST provide an AEAD Algorithm Negotiation record with precisely one algorithm, and a Next Protocol Negotiation record with precisely one next protocol. The data in the Fixed Key Request record must have length twice the key length N of the AEAD algorithm in the AEAD Algorithm Negotiation record. The first N bytes MUST be the C2S Key and the second set of N bytes MUST be the S2C key. Clients MAY use Keep Alive in combination with this record.

This record MUST NOT be sent by a server. A server MAY treat this record as unknown for clients that are not authenticated as described in Section 4.1.

6.6. NTP Server Deny

Record Type Number: To be assigned by IANA (draft implementations: 0x4003) Critical Bit: 0

When provided by a client, indicates a desire to connect to a server other than the server specified in the record. This can be used to ensure a client receives independent NTP servers from one NTS Key Exchange server without having to potentially try multiple times to get a new server.

A client MAY send multiple of these records if desired. The data in the record SHOULD match that given through an NTPv4 Server Negotiation received in an earlier response from the same NTS Key Exchange server.

MUST NOT be sent by a server. Server MAY at its discretion ignore the request from the client and still provide the given server in an NTPv4 Server Negotiation record.

6.7. Authentication Token

Record Type Number: To be assigned by IANA (draft implementations: 0x4005) Critical Bit: 0

When provided by a client, gives a proof of their identity through a pre-shared secret token. This can be used to allow only certain clients, for example pools, to use certain functionality of an NTS key exchange server. In particular, it can be used to prevent misuse of the keep alive mechanism by clients other than the pool, preventing resource exhaustion denial of service attack.

This record MUST be sent before records that may be refused if not properly authenticated. A client MUST NOT send more than 1 of this record. The data in the record should be an ASCII string, previously agreed through an out of scope mechanism.

The Authentication Token record MUST NOT be sent by a server. A server MAY use the record to gate acceptance of other records such as the Keep Alive, Fixed Key Request, List Server Names, Supported Algorithm List and Supported Next Protocol List records. A server supporting this record MUST support keys of length at least 64 characters. Keys SHOULD be chosen such that they have at least 128 bits of entropy.

7. Security Considerations

7.1. Pool’s position

In the pool design presented above, the pool effectively acts as a man in the middle between the user and the ultimate time source during the NTS Key Exchange portion of the session. This means that the pool has access to the key material of these sessions. Although this is a small additional risk, we consider this acceptable because the pool could already always assign sessions for a user to time servers it controls anyway.

The fact that the pool also gets access to key material makes it less advisable to have a pool as a time source for another pool, as this increases the number of actors with access to the key material even further.

The design above does avoid sharing key material between all time sources. As a consequence, a time source in the pool will not be able to break confidentiality or authenticity of traffic with other time sources of the pool. Furthermore, any traffic directly with the time source has no key material involved that is known to the pool.

It must be noted that clients need to trust the pool to check the TLS certificates of the time sources. It is imperative that the pool does this correctly, and that it has a trusted source of time to be able to do revocation checks.

7.2. Keep alive and denial of service attack risk

The Keep Alive NTS record allows a client to keep an NTS key exchange connection open for significantly longer than usual. If arbitrary clients were allowed to do this, they could use it trivially run a server out of resources such as file descriptors. It is therefore important that public servers restrict keeping connections alive to a limited set of trusted clients. The suggested mechanism for doing this is to use TLS client authentication for these clients.

7.3. Error handling

To avoid giving multiple time sources access to the key material of the end user, it is important that the keys extracted from the TLS session between the user and the pool are sent to at most one time source. If an error occurs after sending the Fixed Key Request record, either with the TLS connection between the pool and the time source, or by being explicitly reported by the time source to the pool, the pool SHOULD return an error to the user. Retrying with a different time source during the same TLS session may unintentionally leave the user vulnerable to the operator of the originally selected time source.

8. IANA Considerations

IANA is requested to allocate the following entries in the Network Time Security Key Establishment Record Types registry [RFC8915]:

Record Type Number	Description	Reference
[[TBD]]	Keep Alive	[[this memo]] Section 6.1
[[TBD]]	Supported Next Protocol List	[[this memo]] Section 6.2
[[TBD]]	Supported Algorithm List	[[this memo]] Section 6.3
[[TBD]]	List Server Names	[[this memo]] Section 6.4
[[TBD]]	Fixed Key Request	[[this memo]] Section 6.5
[[TBD]]	NTP Server Deny	[[this memo]] Section 6.6
[[TBD]]	Authentication Token	[[this memo]] Section 6.7

Table 1

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC8915] Franke, D., Sibold, D., Teichel, K., Dansarie, M., and R. Sundblad, "Network Time Security for the Network Time Protocol", RFC 8915, DOI 10.17487/RFC8915, September 2020, <https://www.rfc-editor.org/rfc/rfc8915>.

9.2. Informative References

[Pool] "NTP Pool website", n.d., <https://www.ntppool.org>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/rfc/rfc5905>.

Acknowledgments

The authors thank Marlon Peeters, Ruben Nijveld and Watson Ladd for their input and discussions during the writing of this document.

Authors' Addresses

David Venhoek
Trifecta Tech Foundation
Email: david@tweedegolf.com

Folkert de Vries
Tweede golf B.V.
Email: folkert@tweedegolf.com

Marc Schoolderman
Tweede golf B.V.
Email: marc@tweedegolf.com