

Workgroup: Network Working Group
Internet-Draft: draft-ietf-ntp-chronos-25
Published: 29 August 2023
Intended Status: Informational
Expires: 1 March 2024
Authors: N. Rozen-Schiff

Hebrew University of Jerusalem
D. Dolev
Hebrew University of Jerusalem
T. Mizrahi
Huawei Network.IO Innovation Lab
M. Schapira
Hebrew University of Jerusalem

A Secure Selection and Filtering Mechanism for the Network Time Protocol with Khronos

Abstract

The Network Time Protocol version 4 (NTPv4), as defined in RFC 5905, is the mechanism used by NTP clients to synchronize with NTP servers across the Internet. This document describes a companion application to the NTPv4 client, named Khronos, which is used as a "watchdog" alongside NTPv4, and provides improved security against time shifting attacks. Khronos involves changes to the NTP client's system process only. Since it does not affect the wire protocol, the Khronos mechanism is applicable to current and future time protocols.

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 <https://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 1 March 2024.

Copyright Notice

Copyright (c) 2023 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

- [1. Introduction](#)
- [2. Conventions Used in This Document](#)
 - [2.1. Terms and Abbreviations](#)
 - [2.2. Notations](#)
- [3. Khronos Design](#)
 - [3.1. Khronos Calibration - Gathering the Khronos Pool](#)
 - [3.2. Khronos's Poll and System Processes](#)
 - [3.3. Khronos's Recommended Parameters](#)
- [4. Operational Considerations](#)
 - [4.1. Load considerations](#)
- [5. Security Considerations](#)
 - [5.1. Threat Model](#)
 - [5.2. Attack Detection](#)
 - [5.3. Security Analysis Overview](#)
- [6. Khronos Pseudocode](#)
- [7. Precision vs. Security](#)
- [8. Implementation Status](#)
 - [8.1. Implementation 1](#)
 - [8.1.1. Coverage](#)
 - [8.1.2. Licensing](#)
 - [8.1.3. Contact Information](#)
 - [8.1.4. Last Update](#)
 - [8.2. Implementation 2](#)
- [9. Acknowledgements](#)
- [10. IANA Considerations](#)
- [11. References](#)
 - [11.1. Normative References](#)
 - [11.2. Informative References](#)
- [Authors' Addresses](#)

1. Introduction

NTPv4, as defined in RFC 5905 [[RFC5905](#)], is vulnerable to time shifting attacks, in which the attacker changes (shifts) the clock of a network device. Time shifting attacks on NTP clients can be based on interfering with the communication between the NTP clients and servers or compromising the servers themselves. Time shifting

attacks on NTP are possible even if NTP communication is encrypted and authenticated. A weaker machine-in-the-middle (MitM) attacker can shift time simply by dropping or delaying packets, whereas a powerful attacker, who has full control over an NTP server, can do so by explicitly determining the NTP response content. This document introduces a time shifting mitigation mechanism called Khronos. Khronos can be integrated as a background monitoring application ("watchdog") that guard against time shifting attacks in any NTP client. An NTP client that runs Khronos is interoperable with [\[RFC5905\]](#)-compatible NTPv4 servers. The Khronos mechanism does not affect the wire mechanism and is therefore applicable to any current or future time protocol.

Khronos is a mechanism that runs in the background, continuously monitoring client clock (which is updated by NTPv4) and calculating an estimated offset which we refer by "Khronos time offset". When the offset exceeds a predefined threshold (specified in [Section 5.2](#)), this is interpreted as the client experiencing a time shifting attack. In this case, Khronos updates the client's clock.

When the client is not under attack, Khronos is passive, allowing NTPv4 to control the client's clock and providing the ordinary high precision and accuracy of NTPv4. When under attack, Khronos takes control over the client's clock, mitigating the time shift, while guaranteeing relatively high accuracy with respect to UTC and precision, as discussed in [Section 7](#).

By leveraging techniques from distributed computing theory for time-synchronization, Khronos achieves accurate time even in the presence of powerful attackers who are in direct control of a large number of NTP servers. Khronos will prevent shifting the clock when the ratio of compromised time samples is below $2/3$. In each polling interval, Khronos client randomly selects and samples a few NTP servers out of a local pool of hundreds of servers. Khronos is carefully engineered to minimize the load on NTP servers and the communication overhead. In contrast, NTPv4, employs an algorithm which typically relies on a small subset of the NTP server pool (e.g., 4 servers) for time synchronization, and is much more vulnerable to time shifting attacks. Configuring NTPv4 to use several hundreds of servers will increase its security, but will incur very high network and computational overhead compared to Khronos and will be bounded by compromised ratio of half of the time samples.

A Khronos client iteratively "crowdsources" time queries across NTP servers and applies a provably secure algorithm for eliminating "suspicious" responses and for averaging over the remaining responses. In each Khronos poll interval, the Khronos client selects, uniformly at random, a small subset (e.g., 10-15 servers) of a large server pool (containing hundreds of servers). While

Khronos queries around 3 times more servers per polling interval than NTP, Khronos's polling interval can be longer (e.g., 10 times longer) than NTPv4, thereby, minimizing the load on NTP servers and the communication overhead. Moreover, Khronos's random server selection may even help to distribute queries across the whole pool.

Khronos's security was evaluated both theoretically and experimentally with a prototype implementation. According to this security analysis, if a local Khronos pool consists of, for example, 500 servers, 1/7 of whom are controlled by an attacker and Khronos queries 15 servers in each Khronos poll interval (around 10 times the NTPv4 poll interval), then over 20 years of effort are required (in expectation) to successfully shift time at a Khronos client by over 100 ms from UTC. The full exposition of the formal analysis of this guarantee is available at [[Khronos paper](#)].

Khronos introduces a watchdog mechanism that maintains a time offset value that is used as a reference for detecting attacks. The time offset value computation differs from the current NTPv4 in two key aspects. First, Khronos periodically communicates, in each Khronos poll interval, with only a few (tens) randomly selected servers out of a pool consisting of a large number (e.g., hundreds) of NTP servers. Second, Khronos computes "Khronos time offset" based on an approximate agreement technique to remove outliers, thus limiting the attacker's ability to contaminate the "time samples" (offsets) derived from the queried NTP servers. These two aspects allow Khronos to minimize the load on the NTP servers and to provide provable security guarantees against both MITM attackers and attackers capable of compromising a large number of NTP servers.

We note that, to some extent, NTS [[RFC8915](#)] could make it more challenging for attackers to perform MITM attacks, but is of little impact if the servers themselves are compromised.

2. Conventions Used in This Document

2.1. Terms and Abbreviations

NTPv4 Network Time Protocol version 4 [[RFC5905](#)].

System process Selection Algorithm and the Cluster Algorithm [[RFC5905](#)].

Security Requirements Security Requirements of Time Protocols in Packet Switched Networks [[RFC7384](#)].

NTS Network Time Security for the Network Time Protocol [[RFC8915](#)].

2.2. Notations

Describing Khronos algorithm, the following notation is used.

Notation	Meaning
n	The number of candidate servers in Khronos pool (potentially hundreds).
m	The number of servers that Khronos queries in each poll interval (up to tens).
w	An upper bound on the distance between any "truechimer" NTP server (as in [RFC5905]) and UTC.
B	An upper bound on the client's clock error rate (ms/sec).
ERR	An upper bound on the client's clock error between Khronos polls (ms).
K	The number of Khronos pool re-samplings until reaching "Panic mode".
H	Predefined threshold for time offset triggering clock update by Khronos.

Table 1: Khronos Notations

The recommended values are discussed in [Section 3.3](#).

3. Khronos Design

Khronos watchdog periodically queries a set of m (tens) servers from a large (hundreds) server pool in each Khronos poll interval, where the m servers are selected from the server pool at random. Based on empirical analyses, to minimize the load on NTP servers while providing high security, the Khronos poll interval should be around 10 times the NTPv4 poll interval (i.e., a Khronos clock update occurs once every 10 NTPv4 clock updates). In each Khronos poll interval, if the Khronos time offset exceeds a predetermined threshold (denoted as H), an attack is indicated.

Unless an attack is indicated, Khronos uses only one sample from each server (avoiding "Clock Filter Algorithm" as defined in section 10 in [\[RFC5905\]](#)). When under attack, Khronos uses several samples from each server, and executes the "Clock Filter Algorithm" for choosing the best sample from each server, with low jitter. Then, given a sample from each server, Khronos discards outliers by executing the procedure described in [Section 3.2](#).

Between consecutive Khronos polls, Khronos keeps track of clock offsets, for example by catching clock discipline (as in [\[RFC5905\]](#)) calls. The sum of offsets is referred to as "Khronos inter-poll offset" (denoted as tk) which is set to zero after each Khronos poll.

3.1. Khronos Calibration - Gathering the Khronos Pool

Calibration is performed at the first time the Khronos is executed, and also periodically, once in a long time (every two weeks). The calibration process generates a local Khronos pool of n (up to hundreds) NTP servers the client can synchronize with. To this end, Khronos makes DNS queries to addresses of NTP pools collect the union of all received IP addresses. The servers in the Khronos pool should be scattered across different regions to make it harder for an attacker to compromise, or gain machine-in-the-middle capabilities, with respect to a large fraction of the Khronos pool. Therefore, Khronos calibration queries general NTP server pools (for example pool.ntp.org), and not only the pool in the client's state or region. In addition, servers can be selected to Khronos pool manually or by using other NTP pools (such as NIST internet time servers).

The first Khronos update requires m servers, which can be found in several minutes. Moreover, it is possible to query several DNS pool names to vastly accelerate the calibration and the first update.

The calibration is the only Khronos part where DNS traffic is generated. Around 125 DNS queries are required by Khronos to obtain addresses of 500 NTP servers which is higher than Khronos pool size (n). Assuming the calibration period is two weeks, the expected DNS traffic generated by Khronos client is less than 10 DNS queries per day, which is usually several orders of magnitude lower than the total daily number of DNS queries per machine.

3.2. Khronos's Poll and System Processes

In each Khronos poll interval the Khronos system process randomly chooses a set of m (tens) servers out of the Khronos pool of n (hundreds) servers and samples them. Note that the randomness of the server selection is crucial for the security of the scheme and therefore any Khronos implementation must use secure randomness implementation such as used for encryption key generation.

Khronos's polling times of different servers may spread uniformly within its poll interval, similar to NTPv4. Servers which do not respond during the Khronos poll interval are filtered out. If less than $1/3$ of the m servers are left, a new subset of servers is immediately sampled, in the exact same manner (called "resampling" process).

Next, out of the time-samples received from this chosen subset of servers, the lowest third of the samples' offset values and highest third of the samples' offset values are discarded.

Khronos checks that the following two conditions hold for the remaining sampled offsets:

- *The maximal distance between every two offsets does not exceed $2w$ (can be verified by considering just the minimum and the maximum offsets).

- *The distance between the offsets average and Khronos inter-poll offset is at most $ERR+2w$.

(where w and ERR are as described in [Table 1](#)).

In the event that both of these conditions are satisfied, the average of the offsets is set to be the "Khronos time offset". Otherwise, resampling is performed. This process spreads Khronos client's queries across servers thereby improving security against powerful attackers (as discussed in [Section 5.3](#)) and mitigating the effect of a DoS attack on NTP servers that renders them non-responsive. This resampling process continues in subsequent Khronos poll intervals until the two conditions are both satisfied or the number of times the servers are re-sampled exceeds a "Panic Trigger" (K in [Table 1](#)), in which case Khronos enters a "Panic Mode".

In panic mode, Khronos queries all the servers in its local Khronos pool, orders the collected time samples from lowest to highest and eliminates the lowest third and the highest third of the samples. The client then averages over the remaining samples, and sets this average to be the new "Khronos time offset".

If the Khronos time offset exceeds a predetermined threshold (H) it is passed on to the clock discipline algorithm in order to steer the system time (as in [\[RFC5905\]](#)). In this case the user and/or admin of the client machine should be notified about the detected time shifting attack, for example by a message written to a relevant event log or displayed on screen.

Note that resampling follows immediately the previous sampling since waiting until the next polling interval may increase the time shift in face of attack. This shouldn't generate high overhead since the number of resamples is bounded by K (after K resamplings, "Panic mode" is in place) and the chances to arrive to repeated resampling are low (see [Section 5](#) for more details). Moreover, in an interval following a panic mode, Khronos executes the same system process which starts by querying only m servers (regardless of previous panic).

3.3. Khronos's Recommended Parameters

According to empirical observations (presented in [\[Khronos paper\]](#)), querying 15 servers at each poll interval (i.e., $m=15$) out of 500

servers (i.e., $n=500$), and setting w to be around 25 ms provides both high time accuracy and good security. Specifically, when selecting $w=25\text{ms}$, approximately 83% of the servers' clocks are at most w -away from UTC, and within $2w$ from each other, satisfying the first condition of Khronos's system process. For similar reason, the threshold for time offset triggering clock update by Khronos (H) should be between w to $2w$ and is selected on default to 30ms. Note that in order to support congested links scenarios, it is recommended to use a higher w value, such as 1 sec.

Furthermore, according to Khronos security analysis, setting K to be 3 (i.e., if after 3 re-samplings the two conditions are not satisfied then Khronos enters "panic mode") is safe when facing time shifting attacks. In addition, the probability of an attacker forcing a panic mode on a client when K equals 3, is negligible (less than 0.000002 for each polling interval).

Khronos's effect on precision and accuracy are discussed in [Section 7](#) and [Section 5](#).

4. Operational Considerations

Khronos is designed in order to defend NTP clients from time shifting attacks while using public NTP servers. As such, Khronos is not applicable for datacenters and enterprises which synchronize with local atomic clocks, GPS devices or a dedicated NTP server (for example due to regulations).

Khronos can be used for devices that require and depend upon time keeping withing a configurable constant distance from UTC.

4.1. Load considerations

One requirement from Khronos is thus not to induce excessive load on NTP servers beyond that of NTPv4, even if widely integrated into NTP clients. We discuss below the possible causes for Khronos-induced load on servers and how this can be mitigated.

Servers in pool.ntp.org are weighted differently by the NTP server pool when assigned to NTP clients. Specifically, server owners define a ``server weight'' (the ``netspeed'' parameter) and servers are assigned to clients probabilistically according to their proportional weight. Khronos (watchdog mode) queries are equally distributed across a pool of servers. To avoid overloading servers, Khronos queries servers less frequently than NTPv4, with Khronos query interval set to 10 times the default NTPv4 maxpoll (interval) parameter. Hence, if Khronos queries are targeted at servers in pool.ntp.org, any target increase in server load (in terms of multiplicative increase in queries or number of bytes per second) is

controlled by the poll interval configuration which was analyzed in [\[Ananke_paper\]](#).

Consider the scenario where an attacker attempts to generate significant load on NTP servers by triggering multiple consecutive panic modes at multiple NTP clients. We note that to accomplish this, the attacker must have man-in-the-middle capabilities with respect to the communication between each and every client in a large group of clients and a large fraction of all NTP servers in the queried pool. This implies that the attacker must either be physically located at a central location (e.g., at the egress of a large ISP) or launch a wide scale attack (e.g., on BGP or DNS) and thereby capable to carry similar and even higher impact attacks regardless of Khronos clients.

5. Security Considerations

5.1. Threat Model

The following powerful attacker, including MitM, is considered: the attacker is assumed to control a subset (e.g., third) of the servers in NTP pools and is capable of fully determining the values of the time samples returned by these NTP servers. The threat model encompasses a broad spectrum of attackers, ranging from fairly weak (yet dangerous) MitM attackers only capable of delaying and dropping packets (for example using the Bufferbloat attack) to extremely powerful attackers who are in control of (even authenticated) NTP servers (see detailed security requirements discussion in [\[RFC7384\]](#)).

The attackers covered by this model might be, for example, (1) in direct control of a fraction of the NTP servers (e.g., by exploiting a software vulnerability), (2) an ISP (or other Autonomous-System-level attacker) on the default BGP paths from the NTP client to a fraction of the available servers, (3) a nation state with authority over the owners of NTP servers in its jurisdiction, or (4) an attacker capable of hijacking (e.g., through DNS cache poisoning or BGP prefix hijacking) traffic to some of the available NTP servers. The details of the specific attack scenario are abstracted by reasoning about attackers in terms of the fraction of servers with respect to which the attacker has adversarial capabilities. Attackers that can impact communications with (or control) higher fraction of the servers, for example all servers, are out of scope. Considering pool size to be thousands across the world, such attackers will most probably be capable of performing far worst damage than time shifting.

Notably, Khronos provides protection from MitM and powerful attacks that cannot be achieved by cryptographic authentication protocols

since even with such measures in place an attacker can still influence time by dropping/delaying packets. However, adding an authentication layer (e.g., NTS [[RFC8915](#)]) to Khronos will enhance its security guarantees and enable the detection of various spoofing and modification attacks.

Moreover, Khronos uses randomness to independently select the queried servers in each poll interval, preventing attackers from exploiting observations of past server selections.

5.2. Attack Detection

Khronos detects time-shifting attacks by constantly monitoring NTPv4's (or potentially any other current or future time protocol) clock and the offset computed by Khronos and checking whether the offset exceeds a predetermined threshold (H). Unless an attack was detected, NTPv4 controls the client's clock. Under attack, Khronos takes control over the clients clock in order to prevent its shift.

Analytical results (in [[Khronos paper](#)]) indicate that if a local Khronos pool consists of 500 servers, 1/7 of whom are controlled by a machine-in-the-middle attacker, and 15 servers are queried in each Khronos poll interval, then success in shifting time of a Khronos client by even a small degree (100 ms), takes many years of effort (over 20 years in expectation). See a brief overview of Khronos's security analysis below.

Khronos's security analysis is briefly described next.

5.3. Security Analysis Overview

Time-samples that are at most w away from UTC are considered "good", whereas other samples are considered "malicious". Two scenarios are considered:

- *Less than $2/3$ of the queried servers are under the attacker's control.

- *The attacker controls more than $2/3$ of the queried servers.

The first scenario, where there are more than $1/3$ good samples, consists of two sub-cases: (i) there is at least one good sample in the set of samples not eliminated by Khronos (in the middle third of samples), and (ii) there are no good samples in the remaining set of samples. In the first of these two cases (at least one good sample in the set of samples that was not eliminated by Khronos), the other remaining samples, including those provided by the attacker, must be close to a good sample (for otherwise, the first condition of Khronos's system process in [Section 3.2](#) is violated and a new set of servers is chosen). This implies that the average of the remaining

samples must be close to UTC. In the second sub-case (where there are no good samples in the set of remaining samples), since more than a third of the initial samples were good, both the (discarded) third lowest-value samples and the (discarded) third highest-value samples must each contain a good sample. Hence, all the remaining samples are bounded from both above and below by good samples, and so is their average value, implying that this value is close to UTC [[RFC5905](#)].

In the second scenario, where the attacker controls more than 2/3 of the queried servers, the worst possibility for the client is that all remaining samples are malicious (i.e., more than w away from UTC). However, as proved in [[Khronos paper](#)], the probability of this scenario is extremely low even if the attacker controls a large fraction (e.g., 1/4) of the n servers in the local Khronos pool. Therefore, the probability that the attacker repeatedly reaches this scenario decreases exponentially, rendering the probability of a significant time shift negligible. We can express the improvement ratio of Khronos over NTPv4 by the ratios of their single shift probabilities. Such ratios are provided in Table [Table 2](#), where higher values indicate higher improvement of Khronos over NTPv4 and are also proportional to the expected time till a time shift attack succeeds once.

Attack Ratio	6 samples	12 samples	18 samples	24 samples	30 samples
1/3	1.93e+01	3.85e+02	7.66e+03	1.52e+05	3.03e+06
1/5	1.25e+01	1.59e+02	2.01e+03	2.54e+04	3.22e+05
1/7	1.13e+01	1.29e+02	1.47e+03	1.67e+04	1.90e+05
1/9	8.54e+00	7.32e+01	6.25e+02	5.32e+03	4.52e+04
1/10	5.83e+00	3.34e+01	1.89e+02	1.07e+03	6.04e+03
1/15	3.21e+00	9.57e+00	2.79e+01	8.05e+01	2.31e+02

Table 2: Khronos Improvement

In addition to evaluating the probability of an attacker successfully shifting time at the client's clock, we also evaluated the probability that the attacker succeeds in launching a DoS attack on the servers by causing many clients to enter a panic mode (and query all the servers in their local Khronos pools). This probability (with the previous parameters of $n=500$, $m=15$, $w=25$ and $K=3$) is negligible even for an attacker who controls a large number of servers in client's local Khronos pools, and it is expected to take decades to force panic mode.

Further details about Khronos's security guarantees can be found in [[Khronos paper](#)].

6. Khronos Pseudocode

The pseudocode for Khronos Time Sampling Scheme, which is invoked in each Khronos poll interval is as follows:

```
counter = 0
S = []
T = []
While counter < K do
    S = sample(m) //gather samples from (tens of) randomly chosen serv
    T = bi_side_trim(S,1/3) //trim lowest and highest thirds
    if (max(T) - min(T) <= 2w) and (|avg(T) - tk| < ERR + 2w) Then
        return avg(T) // Normal case
    end
    counter ++
end
// panic mode
S = sample(n)
T = bi-sided-trim(S,1/3) //trim lowest and highest thirds
return avg(T)
```

7. Precision vs. Security

Since NTPv4 updates the clock at times when no time-shifting attacks are detected, the precision and accuracy of a Khronos client are the same as NTPv4 at these times. Khronos is proved to maintain an accurate estimation of the UTC with high probability. Therefore when Khronos detects that client's clock error exceeds a threshold (H), it considers it as an attack and takes control over the client's clock. As a result, the time shift is mitigated and high accuracy is guaranteed (the error is bounded by H).

Khronos is based on crowdsourcing across servers and regions, changes the set of queried servers more frequently than NTPv4 [[Khronos paper](#)], and avoids some of the filters in NTPv4's system process. These factors can potentially harm its precision. Therefore, a smoothing mechanism can be used, where instead of a simple average of the remaining samples, the smallest (in absolute value) offset is used unless its distance from the average is higher than a predefined value. Preliminary experiments demonstrated promising results with precision similar to NTPv4.

Note that in applications such as multi source media streaming, which are highly sensitive to time differences among hosts, it is advisable to use Khronos at all hosts in order to obtain high precision even in the presence of attackers that try to shift each host in a different magnitude and/or direction. Another more

efficient approach for this cases may be to allow direct time synchronization between one host who runs Khronos to others.

8. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [\[RFC7942\]](#). The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [\[RFC7942\]](#), "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

8.1. Implementation 1

Organization: Hebrew University of Jerusalem

Implementers: Neta Rozen-Schiff, May Yaaron, Noam Caspi and Shahar Cohen

Maturity: Proof-of-Concept Prototype

This implementation was used to check consistency and to ensure completeness of this specification.

8.1.1. Coverage

This implementation covers the complete specification.

8.1.2. Licensing

The code is released under the MIT license.

The source code is available at: <https://github.com/netars/chronos>

8.1.3. Contact Information

Contact Martin Langer: neta.r.schiff@gmail.com

8.1.4. Last Update

The implementation was updated in June 2022.

8.2. Implementation 2

Organization: Network Time Foundation (NTF)

Implementers: Neta Rozen-Schiff, Danny Mayer, juergen perlinger and Harlan Stenn.

Contact Martin Langer: neta.r.schiff@gmail.com

Maturity: in progress (<https://khronos.nwtime.org/>).

9. Acknowledgements

The authors would like to thank Erik Kline, Miroslav Lichvar, Danny Mayer, Karen O'Donoghue, Dieter Sibold, Yaakov. J. Stein, Harlan Stenn, Hal Murray, Marcus Dansarie, Geoff Huston, Roni Even, Benjamin Schwartz, Tommy Pauly, Rob Sayre, Dave Hart and Ask Bjorn Hansen for valuable contributions to this document and helpful discussions and comments.

10. IANA Considerations

This memo includes no request to IANA.

11. References

11.1. Normative References

[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/info/rfc5905>>.

[RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", RFC 7384, DOI 10.17487/RFC7384, October 2014, <<https://www.rfc-editor.org/info/rfc7384>>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.

[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/info/rfc8915>>.

11.2. Informative References

[Ananke_paper] Perry, Y., Rozen-Schiff, N., and M. Schapira, "Preventing (Network) Time Travel with Chronos", 2021, <https://www.ndss-symposium.org/wp-content/uploads/ndss2021_1A-2_24302_paper.pdf>.

[Khronos_paper] Deutsch, O., Rozen-Schiff, N., Dolev, D., and M. Schapira, "Preventing (Network) Time Travel with Chronos", 2018, <https://www.ndss-symposium.org/wp-content/uploads/2018/02/ndss2018_02A-2_Deutsch_paper.pdf>.

Authors' Addresses

Neta Rozen-Schiff
Hebrew University of Jerusalem
Jerusalem
Israel

Phone: [+972 2 549 4599](tel:+97225494599)
Email: neta.r.schiff@gmail.com

Danny Dolev
Hebrew University of Jerusalem
Jerusalem
Israel

Phone: [+972 2 549 4588](tel:+97225494588)
Email: danny.dolev@mail.huji.ac.il

Tal Mizrahi
Huawei Network.IO Innovation Lab
Israel

Email: tal.mizrahi.phd@gmail.com

Michael Schapira
Hebrew University of Jerusalem
Jerusalem
Israel

Phone: [+972 2 549 4570](tel:+97225494570)
Email: schapiram@huji.ac.il