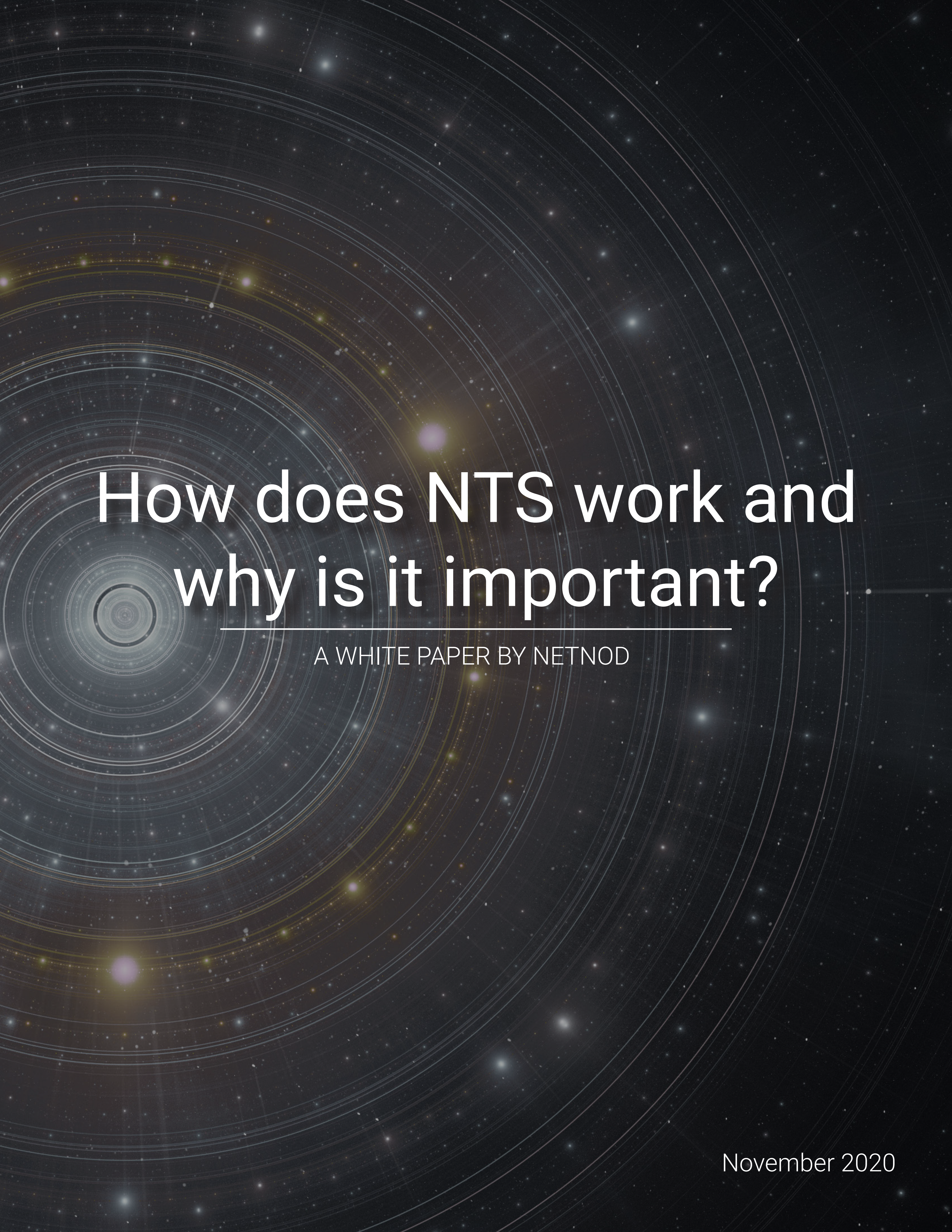


How does NTS work and why is it important?

A WHITE PAPER BY NETNOD

November 2020

Executive Summary

Many of the important security tools that keep us safe online depend on accurate time. But until recently there was no way to ensure that the time you received over the Internet was correct. There was simply no way to tell if you were being fed time from a malicious or trusted source.

This is largely because the current standard for receiving time over the Internet, the Network Time Protocol (NTP), was created in 1985. In those more innocent times, the need to secure NTP, the type of security needed, and how to provide it were less understood.

Over the last 35 years, a range of security flaws and some high-profile attacks have shown that NTP needed significantly improved security. The new Network Time Security (NTS) standard has been designed to fix that.

This white paper gives an overview of NTS and a detailed description of the authentication process that ensures you receive time information from a trusted source. This includes the step-by-step details of the key establishment and time stamping process. We also provide a summary of the elements that enable NTS to scale and to secure NTP against a range of attack vectors.

Table of contents

Introduction

NTP security issues

Unsecured NTP is vulnerable to Man-in-the-Middle (MITM) attacks where a malicious actor sits between you and the NTP server, listens in on the conversation, forges messages and lies to you about time. As a lot of processes are dependent on accurate time, the consequences here can be very serious and can include:

- Incorrect timestamps on logs and transactions which can support fraudulent activities or help disguise other criminal action
- Authentication problems, attacks and issues with authentication security measures
- Issues with cryptographic signatures and establishing encrypted sessions such as Transport Layer Security (TLS)
- DNS Security (DNSSEC) failing to work if the client doesn't have accurate time

NTP vulnerabilities

NTP security threats include: packet manipulation, replay attacks, amplification attacks, and spoofing

What is an amplification attack?

Amplification attacks are a particularly common and damaging form of DDoS attack. They use popular UDP-based protocols, such as NTP or the DNS, where the source IP address can be easily spoofed. In some cases, a small request can generate a huge response which is directed at a third party, the victim, to overwhelm a network or server.

Previous attempts at NTP security

There have been some previous attempts to add security to NTP. Commonly referred to as NTP-AUTH, these attempts did not provide all necessary aspects of security (confidentiality, authentication and integrity including protection against replay attacks).

Moreover, the underlying security mechanism for NTP-AUTH (MD5 or SHA-1) could be considered weak at best. NTP-AUTH is based on secret keys but the key agreement is not standardized. This means that any NTP service provider wanting to use NTP-AUTH has to find their own way to exchange keys with the client out-of-band.

This often requires a potential client to send a letter or fax to get the secret key. The need for a server to store unique secret keys for all clients creates scalability problems: as the number of users grows, the storage requirements and especially the ability to rapidly look up a given key makes the service hard to scale.

Given all these issues, it has long been apparent that a new security mechanism for NTP is needed; one based on modern security mechanisms that could also allow the service to scale to potentially millions of clients.

The NTS standard

NTS has been developed within the Internet Engineering Task Force (IETF). In March 2015, the first Internet-Draft of the NTS standard was published. Over the next five years, the draft went through 28 further iterations until the Internet Draft 'Network Time Security for the Network Time Protocol' was published as a Proposed Standard (RFC8915) in September 2020.

NTS uses modern cryptography to add an important layer of security to NTP. It prevents spoofing and MITM attacks by using authenticated packets. Amplification attacks are prevented by ensuring that request and response packets are always the same size.

NTS is really two protocols: a key establishment protocol, and NTP with some new Extension Fields. The reason for using two protocols is separation of concerns:

1. The seldom used key establishment on top of standard Transport Layer Security (TLS), and
2. The (already existing) low latency UDP-based time synchronisation path.

This means that the existing NTP functionality is the same as before, but the time data can now be authenticated.

The NTS authentication process

The authentication process consists of a key establishment and a timestamp request. The key exchange server typically runs on an ordinary computer, but the slim NTS-enabled NTP server is UDP-based and stateless. It can be served from anycast addresses and can be implemented at the hardware level. The NTP server's state about each client is kept in the cookie provided by the client itself with each request. As there can potentially be hundreds of millions of clients, this is crucial for the smooth operation of a large-scale NTS service.

Since the cryptographic operations in the NTS path are symmetric it is both easier to implement them in hardware and to make them use constant time. This increases the accuracy of the time synchronisation and keeps the slower key exchange outside of the time synchronisation path.

NTS uses Authenticated Encryption using the Advanced Encryption Standard (AES), more specifically what is known as Synthetic Initialization Vector (RFC 5297). This is a block cipher mode of operation providing nonce-based, misuse-resistant authenticated encryption. Using AES-SIV enables the encryption processes described below to add integrity and origin authentication. The consequences for this at a hardware level will be discussed in an upcoming white paper.

The key establishment

The key establishment process works as follows: first, a client initiates a key establishment using the NTS-KE protocol embedded in TLS (see figure 1.)

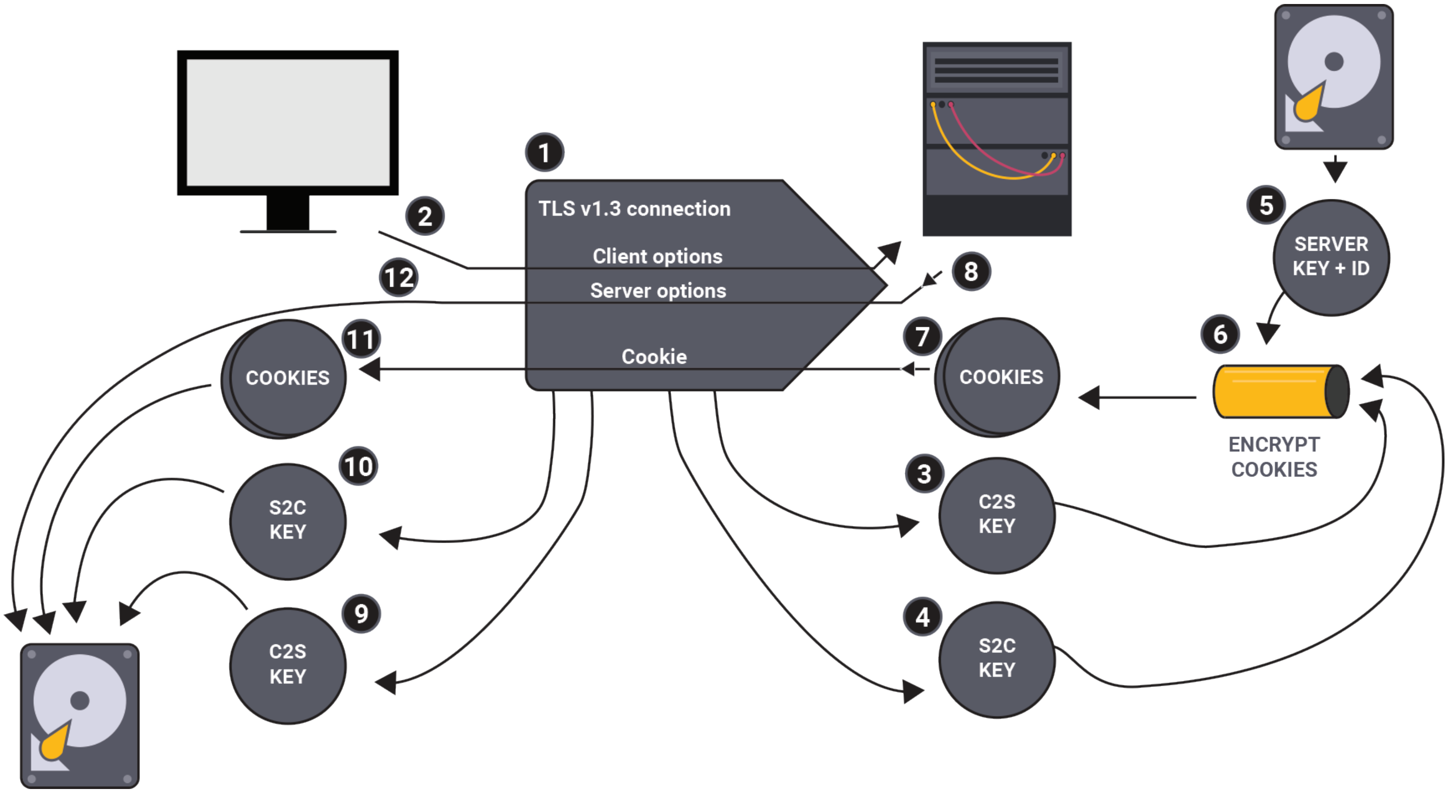


Figure 1: Key establishment. Note: the numbers shown correspond to the text below.

The client makes a TLS v1.3 connection to the server (1). Using TLS means that the client can use normal TLS certificate infrastructure to verify the identity of the server. TLS also provides encryption and ensures the integrity of the data sent over the connection.

The client sends options (2) such as preferred encryption algorithms to the server. The server exports the client-to-server (C2S) key (3) and the server-to-client (S2C) key (4) from the TLS session (RFC 5705). The server then retrieves its server key (5) and uses it to encrypt (6) the C2S key and the S2C key. This encrypted data together with the server key ID and some other information make up a cookie. The server generates eight such cookies (7) which are sent to the client together with any other server options (8) such as where to find the NTS timestamping server.

The client exports the C2S (9) and S2C (10) keys the same way the server did and saves them together with the cookies (11) and server options (12) it received. After the key establishment is done, the server forgets the C2S and S2C keys.

Note: *that the server key is distinct from the C2S and S2C keys. The server key is only known by the server. The S2C and C2S keys are generated during the key establishment, and are unique for each client. This has an important consequence: the only thing the server has to remember is its server key. By storing the C2S and S2C keys encrypted with the server key in the cookie, signing the cookie and then handing the cookie to the client, the server essentially uses the client as the storage for the C2S and S2C keys.*

NTS timestamp request

Next, the client queries the server about time using NTP with NTS extensions.

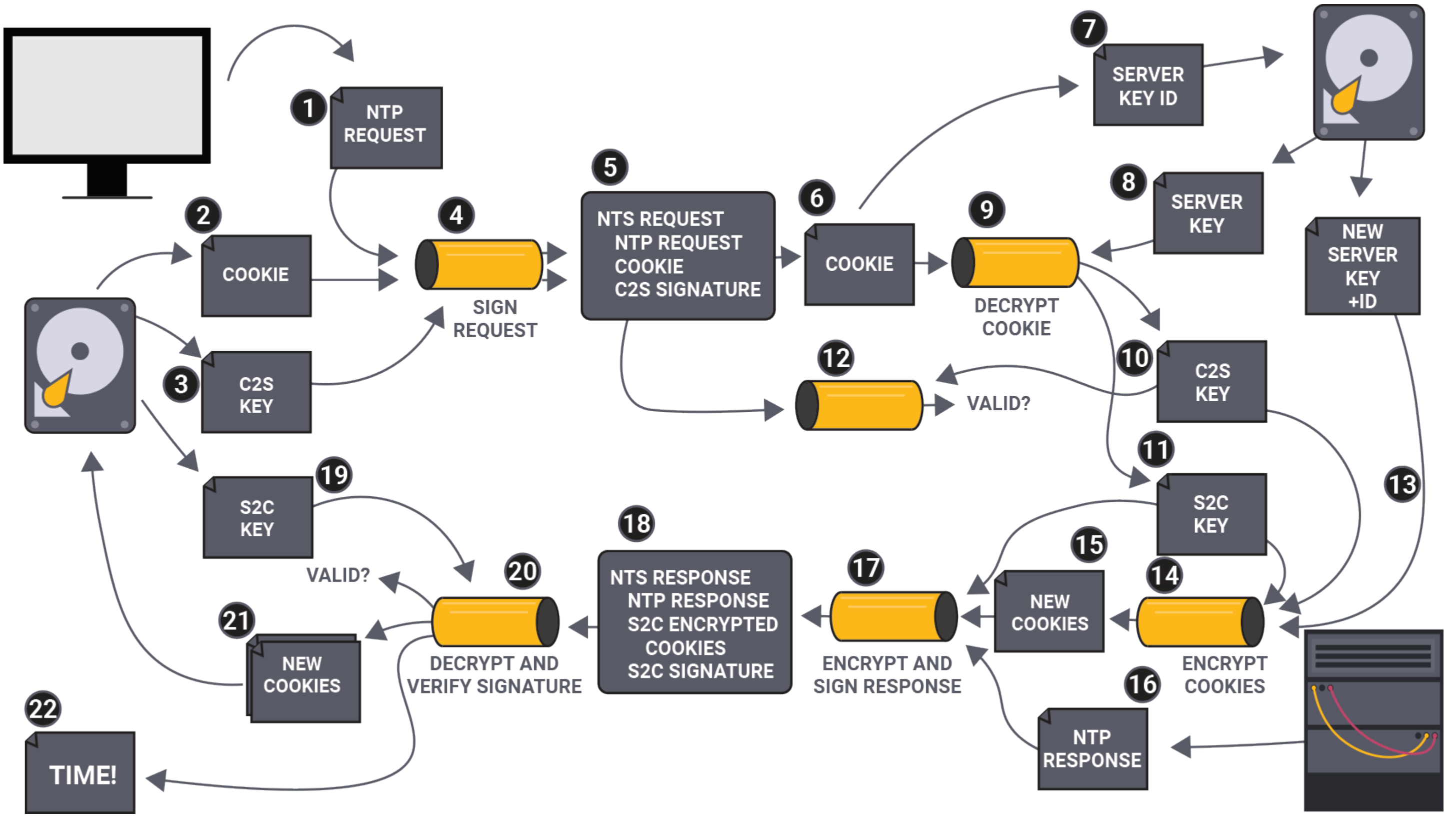


Figure 2: Timestamp request. Note: the numbers shown correspond to the text below.

The client generates a NTP timestamping request (1). It retrieves one of the cookies (2) it received from the server earlier together with the C2S key (3). It then signs (4) the NTP request and the cookie and sends the complete NTS request (5) to the server.

Note that the server can have multiple server keys each with its own ID. The server extracts the cookie (6) from the request. To know which server key to use it extracts the server key ID (7) from the cookie before retrieving the corresponding server key (8). The server key is then used to decrypt the encrypted parts of the cookie (9) and extract the C2S (10) and S2C (11) keys. The C2S key is used to verify the signature (12) of the NTP request.

The server retrieves the most recent server key and ID (13) and uses this to create one or more new cookies which are encrypted (14) using the S2C key. The encrypted cookies (15) together with the NTP response (16) are signed (17) using the S2C key. The complete NTS response (18) is sent to the client.

The client retrieves the S2C key (19) and uses it to decrypt the NTS response and validate the signature (20). It extracts the cookies (21) from the decrypted part of the response and stores them for later use. It can use the time (22) from the NTP response to adjust the time knowing that it was sent from the correct server.

The anatomy of an NTS timestamp request

The NTS timestamp request (shown in (5) in Figure 2 above) consists of the following elements which serve the functions summarised below.

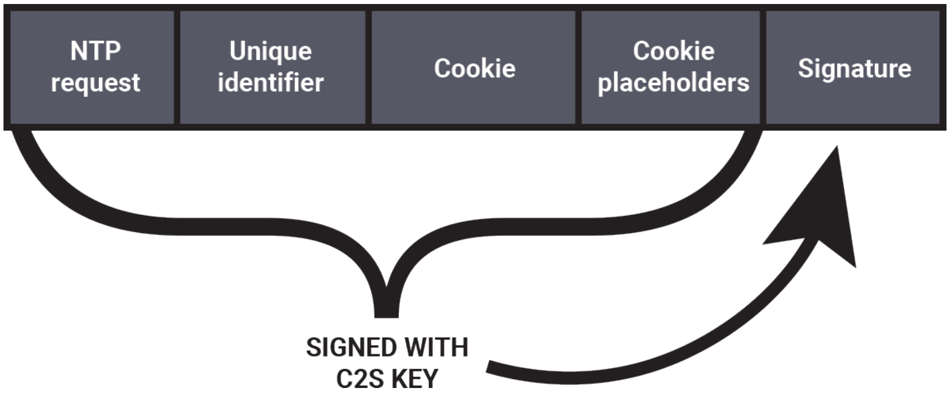


Figure 3: The anatomy of an NTS timestamp request

- The NTP request
- Unique identifier (some random data so the client can match the response to the request and can avoid replay attacks)
- Cookie (one of the cookies provided by the server)
- Optional cookie placeholders
    - A client can request more cookies from the server by adding cookie placeholders
    - The placeholders ensure that the response will not be larger than the request. This prevents amplification attacks
- An "authenticator" which contains a cryptographic signature of the request
    - It can also contain encrypted data, but this is not currently used in NTS
    - The signing is done using the C2S key

The anatomy of an NTS timestamp response

The NTS timestamp response (shown in (18) in Figure 2 above) consists of the following elements which serve the functions summarised below.

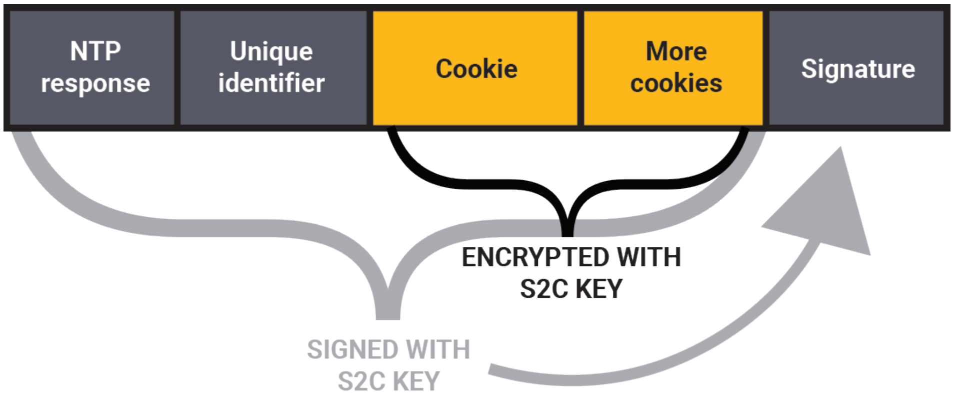


Figure 4: The anatomy of an NTS timestamp response

- The NTP response
- The unique identifier copied from the NTS request
  - The client can use this to match the response to the request
- Authenticator with encrypted data and signature
  - The encrypted portion contains the new cookie(s)
  - The signature signs both the unencrypted and encrypted parts of the response
  - The encryption and the signing are done using the S2C key

The anatomy of a cookie

The cookie is one of the most important parts of NTS and its implementation is central to how a NTS server can scale to a large number of clients. The cookie consists of the following elements which serve the functions summarised below.

Note: since a cookie is opaque to a client, an NTS server implementation is free to format cookies some other way as long as it can somehow extract the C2S and S2C keys from information in the cookie. The description below uses the cookie format recommended in the NTS standard, which is also used in Netnod’s implementation.

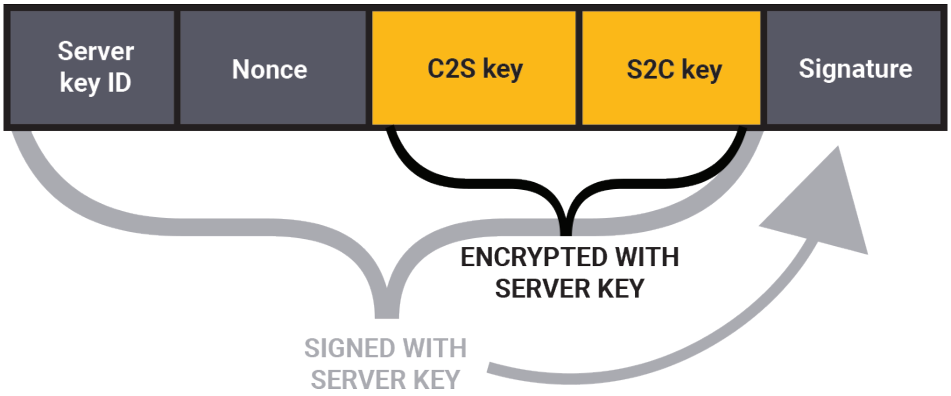


Figure 5: The anatomy of a cookie

- The server can have multiple server keys
    - The Server Key ID in the cookie identifies the server key which was used for the encrypted parts of the cookie
- The nonce is just random data
        - The nonce will make each cookie unique
    - Without this, the encrypted portion of the cookie would be identical for all cookies using the same server key and having the same C2S and S2C keys
    - This is what ensures “unlinkability”. As long as a client does not reuse a cookie, it is impossible for a “man in the middle” to link two NTS requests together and to tell if they came from the same client.
- The C2S and S2C keys are encrypted with the server key
    - Only the server knows this key
    - Only the server can decrypt the S2C and C2S keys
- Everything in the cookie is signed with the server key
    - The server can verify that the cookie was actually created by the server

More information

Netnod’s NTS-enabled NTP service is freely available to the public. More information is available at:
https://www.netnod.se/time-and-frequency/network-time-security

Netnod has published a HOWTO explaining how to set up an NTS client and to connect to Netnod’s NTS servers at:
https://www.netnod.se/time-and-frequency/how-to-use-nts

Some current NTP clients supporting NTS (two of which were written by Netnod staff) include:

ntpsec (written by Eric Raymond)
https://gitlab.com/NTPsec/ntpsec

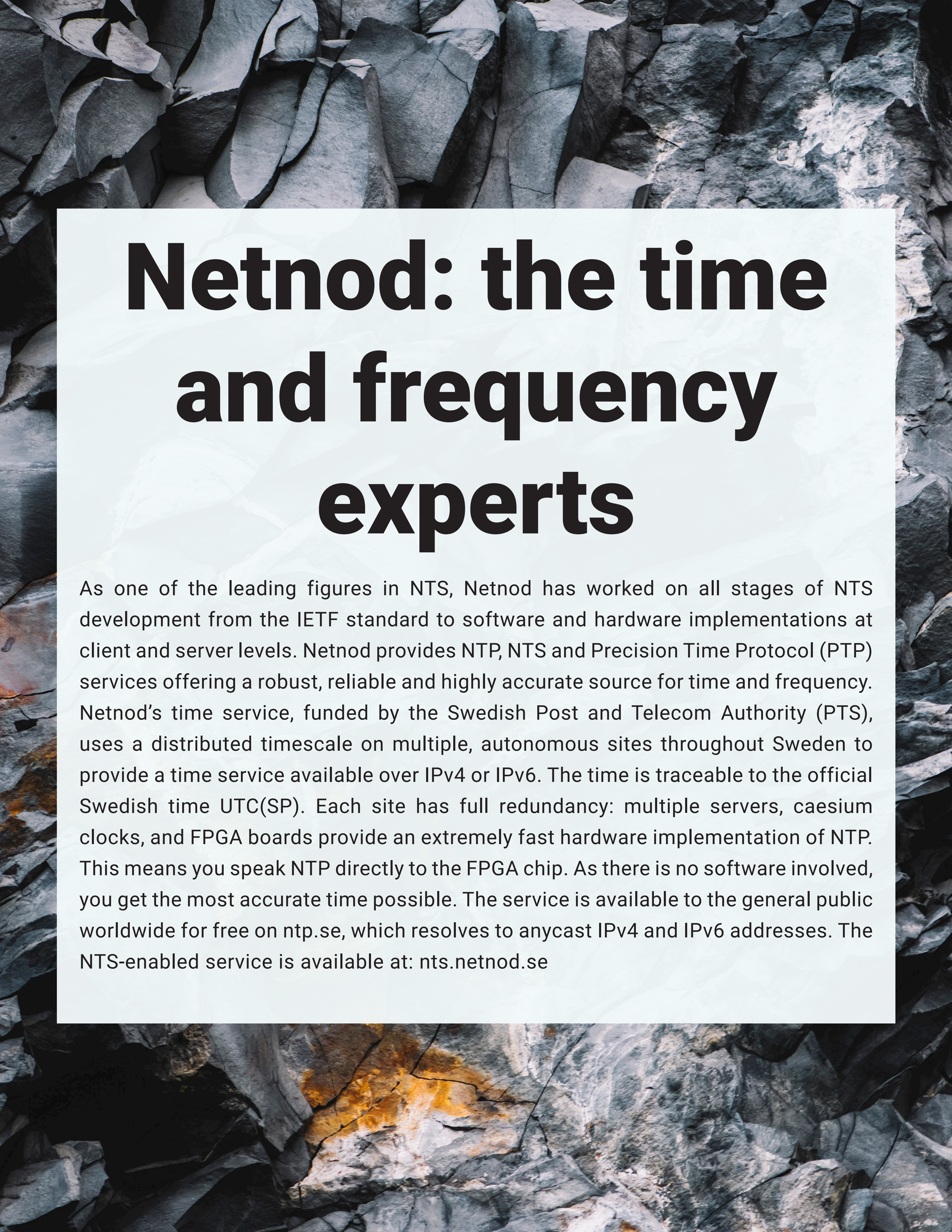
Chrony (written by Richard Curnow, currently maintained by Miroslav Lichvar)
https://chrony.tuxfamily.org/

A Python implementation (written by Christer Weinigel, Netnod)
https://github.com/Netnod/nts-poc-python

A Go implementation (written by Michael Cardell Widerkrantz (Netnod), Daniel Lublin and Martin Samuelsson)
https://gitlab.com/hacklunch/ntsclient

Netnod’s FPGA implementation has been running since the summer of 2020. More information about the hardware implementation of NTS will follow in an upcoming white paper.

This is an open source licensed project using the BSD 2 clause (https://opensource.org/licenses/BSD-2-Clause).
The public repository for the old FPGA_NTP_SERVER can be found at:
https://github.com/Netnod/FPGA_NTP_SERVER



Netnod: the time and frequency experts

As one of the leading figures in NTS, Netnod has worked on all stages of NTS development from the IETF standard to software and hardware implementations at client and server levels. Netnod provides NTP, NTS and Precision Time Protocol (PTP) services offering a robust, reliable and highly accurate source for time and frequency. Netnod's time service, funded by the Swedish Post and Telecom Authority (PTS), uses a distributed timescale on multiple, autonomous sites throughout Sweden to provide a time service available over IPv4 or IPv6. The time is traceable to the official Swedish time UTC(SP). Each site has full redundancy: multiple servers, caesium clocks, and FPGA boards provide an extremely fast hardware implementation of NTP. This means you speak NTP directly to the FPGA chip. As there is no software involved, you get the most accurate time possible. The service is available to the general public worldwide for free on ntp.se, which resolves to anycast IPv4 and IPv6 addresses. The NTS-enabled service is available at: nts.netnod.se