A Network Time Security Based Automatic Key Management for PTPv2.1

Martin Langer, Kai Heine, Dieter Sibold and Rainer Bermbach

Full Citation of the original article published by IEEE:

M. Langer, K. Heine, D. Sibold and R. Bermbach, "A Network Time Security Based Automatic Key Management for PTPv2.1," 2020 IEEE 45th Conference on Local Computer Networks (LCN), Sydney, Australia, 2020, pp. 144-153, doi: 10.1109/LCN48667.2020.9314809.

Available at:
https://doi.org/10.1109/LCN48667.2020.9314809

A Network Time Security Based Automatic Key Management for PTPv2.1

Martin Langer
Ostfalia University of Applied Sciences
Wolfenbüttel, Germany
mart.langer@ostfalia.de

Kai Heine
Ostfalia University of Applied Sciences
Wolfenbüttel, Germany
ka.heine@ostfalia.de

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

Rainer Bermbach
Ostfalia University of Applied Sciences
Wolfenbüttel, Germany
r.bermbach@ostfalia.de

***Abstract*—The PTPv2.1 standard provides new protection mechanisms to ensure the authenticity and integrity of PTP messages. However, the distribution of the necessary security parameters is not part of the specification. This paper proposes a simple and practical approach for the automated distribution of these parameters by using a key management system that enables the Immediate Security Processing in PTP. It is based on the Network Time Security protocol and offers functions for group management, parameter updating and monitoring mechanisms. A Proof-of-Concept implementation provides initial results of the resources required for the key management system and its use.**

Keywords—IEEE 1588, PTPv2.1, NTS, Key Management

I. Introduction

In many areas, exact time synchronization of devices is important for a correct operation. However, areas such as the financial sector, telecommunications or electrical power distribution depend on accuracies that only the Precision Time Protocol (PTP) can provide. It uses packet-based networks and achieves synchronization accuracies in the nanosecond range. The first version of PTP was released in 2002 [1] and was revised in 2008 as PTPv2 [2]. The current PTP version 2.1 [3], which is backwards compatible to PTPv2 and offers several functional enhancements.

For a long time, little attention was paid to the protection of time information. The Annex K of PTPv2 [2] described an experimental approach to the protection of PTP messages for the first time. But this was not applied in practice, also because of depicted weaknesses in the protocol flow in later analyses [4] [5] [6]. To acknowledge the increased demand on secured time transfer, the specification of the new PTP version 2.1 incorporates a normative Type-Length-Value (TLV) extension for authentication, which protects the integrity and authenticity of PTP packets. Nevertheless, PTPv2.1 does not specify the necessary key distribution mechanism and only suggests possible approaches in its new Annex P (see Chapter II). These proposals base upon existing key distribution protocols, which are not designed for PTP and therefore need to be modified or extended. Thus, no straightforward solution for the realization of such a key management system exists.

To address this gap, this paper proposes a simply kept automatic key management (KM) for PTP version 2.1 based on the Network Time Security Protocol (NTS) [7]. NTS is designed for securing time protocols such as the Network Time Protocol (NTP) [8]. Due to the development work on NTS back then, it was not yet considered in the current PTP standard as a possible key management method. The concept presented here extends the NTS protocol in such a way that it can also be used for key distribution in PTP networks. Thus, secure PTP connections as well as secure NTP connections (e.g. as a so-called watchdog mechanism for PTP [9]) are realizable using a common key management system.

Similar to the Group Domain of Interpretation protocol (GDOI) [10] – one of the approaches suggested in Annex P – our proposal uses a basic group management to secure PTP multicast connections. It supports a simple group control, offers a cyclic update of security parameters and works with the PTPv2.1 concept of *Immediate Security Processing* (see II.A.1). Since our approach is directly oriented towards PTP, the protocol design does not contain any unnecessary complexities and is easy to implement. This is an advantage in comparison to GDOI, which is aligned to IPsec and cannot be used directly for PTP. As with GDOI, our NTS-based multicast approach also has limitations for PTP unicast connections (e.g. scalability), which are discussed in this paper.

In the further course of this paper, Chapter II presents basic information about PTP security and the operation of the Network Time Security protocol. Subsequently, Chapter III provides a brief history of the progress of PTP security and the related work. Chapter IV describes the proposed key management in a short form, explains the basic functions and compares the approach with GDOI. The security considerations are then discussed in Chapter V. The following Chapter VI depicts the automatic key management in detail and describes some specific features of the method. The results of a first Proof-of-Concept (PoC) implementation as well as the performance comparison of the feasible MAC algorithms are shown in Chapter VII. The conclusion and the further course of action are finally discussed in Chapter VIII.

II. Preliminaries

This chapter explains the most important security aspects of the current PTP version 2.1 and gives an overview of the functionality of the Network Time Security protocol.

A. New PTP Security Mechanisms

The security concepts in Precision Time Protocol v2.1 are separated into four so-called Prongs [3], which can be applied individually or together to protect the PTP messages and the PTP network.

1) Prong A: Integrated Security Mechanism: Prong A describes a built-in mechanism of PTP to ensure the authenticity and integrity of the PTPv2.1 messages. This mechanism consists of three components: key management, security processing method and AUTHENTICATION TLV.

The AUTHENTICATION TLV (AuthTLV) is usually located at the end of a PTP message and contains various security parameters as well as an Integrity Check Value (ICV). Together with the Security Policies (SP), the contained security parameters allow a recipient to identify the so-called Security Association (SA). The SA contains all necessary information to construct the AuthTLV and recalculate the ICV. The ICV is generated over the entire PTP message including the AuthTLV omitting the ICV field. If several protection mechanisms are used simultaneously, multiple AuthTLVs can also be present in a PTP-message.

The protection mechanisms are distinguished into *Immediate Security Processing* and *Delayed Security Processing*. With the immediate variant, the necessary security parameters and keys are transmitted upfront, before the secure PTP communication takes place. This allows all devices that know these parameters to immediately check incoming PTP messages. It also enables Transparent Clocks (TC) to modify correction fields in the PTP messages and protect them subsequently. PTPv2.1 proposes the GDOI protocol as a key management system, which allows for extended group control.

With the delayed variant, the key required for verifying the messages is disclosed by the master a posteriori. In this method the time is divided into intervals of fixed duration. All messages that are transmitted in one of these time slots are secured with the respective interval key. The disclosure of the key used takes place in a later interval. The Timed Efficient Stream Loss-Tolerant Authentication Protocol (TESLA) [11], which was designed for securing broadcast messages, represents such a key management system.

The choice of the security processing method is application-specific and can be defined in PTP profiles. The protocols suggested by PTPv2.1 for this purpose (GDOI and TESLA) are recommendations and can be replaced by other methods as long as they permit the construction of the AuthTLV. In general, the PTPv2.1 specification allows the use of both manual and automatic key management systems. In a manual KM system, the security parameters are transferred to the PTP devices upfront and are usually static. While this is feasible for small PTP networks, it is not a solution for larger networks due to scaling issues. In addition, the maintenance effort increases when parameters and keys need to be updated. An automatic KM system solves these problems and enables a much better administration of the PTP network. However, PTPv2.1 does not define a specific key management system nor the associated communication or message structure.

2) Prong B: External Transport Security Mechanism: Another security strategy is the use of external mechanisms like MACsec [12] or IPsec [13]. Both approaches provide authenticity and message integrity and optionally allow the encryption of the higher protocol layers. MACsec works on layer 2 of the Open Systems Interconnection (OSI) model and also protects protocols such as the Address Resolution Protocol (ARP) and the Dynamic Host Configuration Protocol (DHCP), but is restricted to switches or end-to-end connections. IPsec works on layer 3 and allows tunneling across networks, but due to the protocol stack it cannot be used in conjunction with the 802.3 mode (Ethernet) in PTP.

3) Prong C: Architectural Mechanisms: This prong provides guidance on how to protect a PTP network and provides solutions to mitigate denial of service attacks (DoS) by increasing the redundancy. This includes additional time sources and Grandmaster Clocks as well as redundant network paths.

4) Prong D: Monitoring and Management: Another part is monitoring the network to detect attacks or problems in the infrastructure. This includes unexpected offset jumps or large changes in peer-to-peer (P2P) measured PTP link delays. The use of a watchdog mechanism [9] can prevent or mitigate the consequences of such delay attacks.

B. Network Time Security (NTS)

Even the widely used Network Time Protocol (NTP) [8] offered insufficient security mechanisms for a long time. One of them is the older and still secure symmetric key approach, which is unfortunately not scalable and therefore futile in typical NTP use cases. Although this problem was subsequently solved with the Autokey [14] method, an analysis in 2012 revealed other serious design flaws of that procedure [15]. For this reason, the Network Time Security (NTS) [7] protocol was developed to provide the necessary security functions. NTS currently focuses on NTP, but is designed in such a way that it can be extended to other time protocols.

An NTS-secured communication generally consists of three parts: The Transport Layer Security v1.3 (TLSv1.3) [16] handshake, the parameter negotiation and the secured time transmission. Together, TLS connection and parameter negotiation form the NTS Key Establishment protocol (NTS-KE) and can act as a key management system (blue boxes in Figure 1). In the first phase (Phase 1.1), TLSv1.3 enables peer

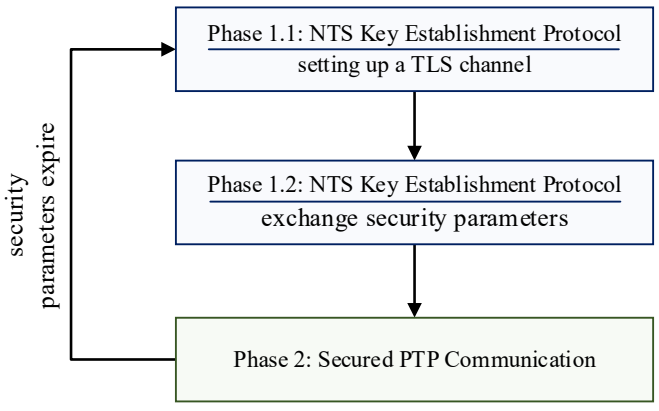

Fig. 1. Phases in the establishment of a secured time transfer

authentication by certificates and provides authenticity, message integrity and confidentiality of following data transmitted over the TLS channel. The messages subsequently exchanged via TLS (Phase 1.2) depend on the time protocol to be protected. These always consist of a set of Records, which each contains necessary information (e.g. protocol type, algorithms, IDs, ...). Since PTP has different requirements than NTP, new Records must be specified that follow the typical TLV Record structure defined in NTS. Chapter VI describes in detail which existing Records can be used for PTP and which new ones are introduced. After transmission of these data the TLS channel is closed and the secured time transfer begins.

While the message exchange in the first phase of NTS (Phase 1.1 and 1.2) is applicable to both NTP and PTPv2.1, the second phase addresses the time protocol to be secured itself. In NTP, extension fields are used to transfer encrypted security parameters in the form of cookies between client and server. These are used also to protect the NTP messages, which allows the server to be stateless and ensures key freshness and untraceability of the client. But this procedure is not useful for PTP, since both the communication structure and the security requirements differ. On the one hand, the cookie method is not applicable to multicast connections, while on the other hand tracking protection and statelessness are not required. However, for PTPv2.1 the Prong A approach is a reasonable solution. The PTP integrated security mechanism already describes the general content and structure of the AuthTLV and the NTS-KE protocol can provide the security parameters needed. If the parameters expire during the protected time transfer operation or other problems occur, the NTS-KE protocol can be executed again.

III. Related Work

This section provides a brief overview of the security developments since the last PTP version 2.0 in 2008 and addresses the advantages, as well as the limitations of current solutions for PTPv2.1.

A. State of the Art

Since IEEE 1588-2008 (PTPv2) [2] the security aspect gains more and more attention. Precision Time Protocol v2.1 already provided a first experimental security mechanism, Annex K, which offered source authentication, message integrity and protection against replay attacks. However, Annex K was not used in practice and showed some design flaws [4] [5] [6]. It proposed a three-step authentication process with a challenge-request-response mechanism used in the start-up phase. This method works, but is inefficient and can be replaced by a one-way authentication. Also, the sequence number may be too short to effectively protect against replay attacks. Furthermore, the distribution of keys is complex and could be improved. Another point is the Keyed-Hash Message Authentication Code (HMAC) that is used for message protection, which is rather unsuitable in one-step mode where on-the-fly protection of messages may be necessary.

Several publications followed, which presented suggestions for improvement of Annex K and alternative key distribution mechanisms like GDOI and TESLA. Naiara Moreira et al. summarized these in their paper [4] and compared the characteristics of the different approaches. In addition to these methods for securing PTP messages, they also examined further protection mechanisms such as MACsec and IPsec (Prong B) in their paper. NTS was also a part of this investigation, which at that time still included TESLA as an approach in its draft [17]. In the specification phase of PTPv2.1, these proposals led to an update of the security measures now defined in the clause 16.14 and the new Annex P [3]. Clause 16.14 defines the normative AUTHENTICATION TLV for securing PTP messages (Prong A). Annex P describes in general terms the four prongs (see Chapter II.A) and potential key management systems (Prong A: GDOI and TESLA). The previous Annex K was thus obsolete and therefore removed. A further approach followed from Kemparaj and Kumar, who deals with the full encryption of PTP messages [18] without an external key management system. However, this approach was not further considered in the PTPv2.1 specification. Also the current NTS protocol [7] was not mentioned in PTPv2.1 as a possible key management system due to its draft status at the time.

The functionality of clause 16.14 (Prong A: PTP Integrated Security) has already been confirmed by two implementations. Ezzeldin Shereen et al. present in [19] a Linux PTP implementation that examines the two procedures Immediate Security Processing and Delayed Security Processing (unauthenticated). Dragos Maftei et al. present a PTPd-based implementation [20], which also supports immediate and delayed security processing. Both implementations use a manual key management system instead of GDOI (immediate processing) or TESLA (delayed processing).

B. Limitations of GDOI and TESLA

The application of GDOI [10] requires, like in the NTS-based proposal, a phase-one protocol. Therefore, it uses the Internet Key Exchange v2 (IKEv2) protocol [21], which is standardized and available like GDOI. However, both are designed for IPsec, which makes a direct use in PTPv2.1 difficult and requires additional specification effort. GDOI's advantage lies among other things in the immediate integrity check of PTP messages and the support of Transparent Clocks. However, the disadvantage is the lower level of source authentication, because every trusted member is able to modify the messages in his own group. More details are discussed in chapter IV.B comparing GDOI to our own NTS-based proposal.

On the other hand, TESLA offers a strong source authentication to the master, so that group members cannot change messages. However, the use of TESLA in a PTP context is very difficult, because every PTP instance which could become a master must be able to operate as a TESLA server. Slaves are also burdened additionally, because the verification of PTP messages is delayed and the PTP-messages have to be stored temporarily. Furthermore, TESLA requires a phase-one protocol for the bootstrapping mechanism (not defined) as well as an already time-synchronous PTP node (in the seconds range). It does not provide any group management functions and does not support Transparent Clocks. Moreover, TESLA can be broken with delay attacks in a short time [22] and needs a secured watchdog mechanism [9] to prevent this.

IV. Key Management for PTP – Overview

Our key management system combines the NTS-KE protocol with the PTP Integrated Security Mechanism. It enables the simple and fast integration of an automatic key management system (NTS-KM) into PTP networks. Based on the Immediate Security Processing of messages, this KM also allows the formation of groups (e.g. based on PTP domains). These can be formed by an administrator at the PTP nodes or on the KM server side. Using a cyclic update process, security policies, keys, and other parameters can be changed during operation without interrupting the PTP communication. Moreover, the approach enables simple group control, to exclude PTP nodes from a group or assign them to other groups. In addition, the protocol can be extended to include further functionalities.

The following section briefly describes the steps that a PTP node must perform to establish a secured communication and how the KM server works. Then the approach depicted here is compared to a GDOI-based solution. For a more detailed description of the KM procedure and configuration, also see Section VI.

A. KM-Protocol Overview and Securing Process

The pre-configuration of the KM server and the PTP node is the first step to secure the communication. Both sides require certificates to check the authenticity of each other.

Any PTP node that wants to be part of a secured PTP network first starts by establishing a TLSv1.3 connection to the KM server (see Figure 2). The host name or the IP address of the server is given to the PTP node in advance, e.g. by using a configuration file. The TLS handshake authenticates the KM server and authorizes the PTP node to join the secure PTP network. In addition, all data transmitted via the TLS channel are secured and encrypted.

The PTP node now sends a *GroupRequest* message over the established TLS channel. This message simply consists of a sequence of data blocks (so-called Records) in a Type-Length-Value (TLV)-like format. Each block transfers specific parameters and is easy to parse without special decoders. The request sent by the PTP node contains the wish to join a specific group or, if the authorization allows for it, multiple different groups. The group selection is derived from the configured *PTP domain* and *sdoId* of the PTP node. The domain permits the separation of PTP network groups, while the sdoId enables the distinction between PTP profiles. This prevents complications when different profiles use the same PTP domain.

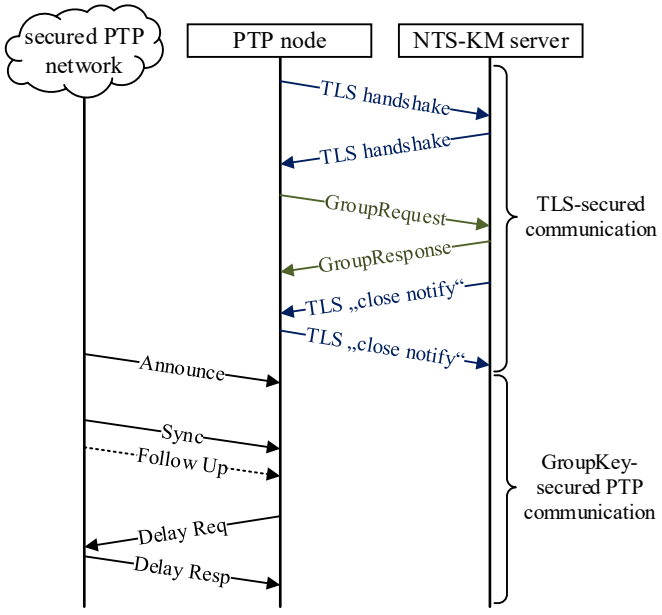

Fig. 2. A PTP node joins a secured PTP network after SP/SA exchange

After reception of the *GroupRequest* the KM server first checks whether joining the group is allowed or whether this was prohibited by restrictions set by the administrator. The KM server stores the decision in a *GroupResponse* message that is structurally identical to the GroupRequest. If access is granted, the security policies and security parameters of the group are also included. Depending on the security policy requirements, different security parameters (Security Associations) can be used for different PTP messages. The message also contains information about the validity period of these parameters and when the PTP node can start to retrieve a new set of parameters from the KM server. In addition, if new parameters are already available for the following validity period, the KM server embeds them in the GroupResponse, too. In this way, PTP nodes can use the new parameters immediately after the expiration of the current ones without interruption of the PTP communication. After sending the GroupResponse message, the KM server also ends the TLS connection correctly with a *close notify* message. The communication with the KM server is thus completed and the PTP node is able to join the secured PTP communication.

To protect or check a PTP message, the PTP node first reads the Policy Limiting Fields (PLF) from the PTP header of the respective message. Among other fields, these include the *domainNumber*, the *sourcePortID* and the *messageType*. If a PTP device manages different security policies (e.g. for different ports), the PLF can be used to identify the correct SP. The SP then provides information about *which* PTP messages are to be secured. If a protection is required for the message type, the SP can be used to identify the corresponding Security Association, which specifies *how* the protection is to be performed. Using the parameters contained in the SA, the construction of the AuthTLV including the calculation of the Integrity Check Value is possible now. Both are appended to the end of the message to be secured. A recipient of this message can also use the AuthTLV to identify the matching SA and thus calculate and check the ICV.

B. Comparison to GDOI

Both the NTS-based key management and the GDOI-based approach realize the Immediate Security Processing of messages and allow the administration of groups.

In our approach we aim for a solution that is easy to implement and easy to employ. The NTS-KM has a very simple structure and uses TLSv1.3 as *phase one* protocol. TLS libraries (e.g. OpenSSL or WolfSSL) are widely available and mostly easy to apply. The GroupRequest/GroupResponse messages are quickly implemented as well and do not require special decoders. The functional range is kept lean and offers periodic parameter updates and group control/modifications. The KM message exchange takes place exclusively via unicast messages

and always starts from the PTP node. The configuration of the groups can be determined both by the PTP node and the KM server. Furthermore, the NTS-KM can also be used to secure NTP messages in order to establish additional redundancies (Prong C). In addition, NTS-secured NTP can be combined with the Watchdog principle described in [9] to detect delay attacks in multicast connections and provide a monitoring mechanism (Prong D). This controls the secured PTP protocol and executes the NTS-secured NTP Watchdog protocol only when necessary.

It is obvious that using an extended NTS key management for PTPv2.1 is an uncomplicated, but reliable solution to provide the necessary keys for secured PTP communication.

GDOI, on the other hand, is strongly oriented towards IPsec and, due to the protocol design, can only be used with modifications in PTPv2.1. Similar to NTS-KM, GDOI also requires a *phase one* protocol in order to transmit messages securely. Here, the use of the Internet Security Association and Key Management Protocol (ISAKMP) [23] or the Internet Key Exchange Protocol Version 2 (IKEv2) [21] is defined. But for GDOI, ISAKMP as well as IKEv2 free libraries are rarely available. Parameter and group adjustments can be performed immediately (using the GROUPKEY-PUSH protocol) and do not require a cyclic update process. The transmission of these data is usually done by multicast and thus initiated by the GDOI key management server. Group forming is not a part of GDOI.

V. Security Considerations

This chapter focuses on the aspects of authentication using certificates, data exchange with the NTS-KM and the secure PTP communication.

A. Certificates and Trusted Anchor

The authentication is mainly based on certificates issued by a trusted Certificate Authority (CA) that are utilized during the TLS handshake. In classical TLS applications only servers are required to have them. With our approach, the PTP nodes also need certificates to allow only authorized and trusted devices to get the group key and join a secure PTP network (see also VI.A). The verification of a certificate always requires a loose time synchronicity, because they have a validity period. This, however, reveals the well-known chicken-and-egg problem, since secure time transfer itself requires valid certificates. Furthermore, some kind of Public Key Infrastructure (PKI) is necessary, which is conceivable via the Online Certificate Status Protocol (OCSP) as well as offline via root CA certificates.

B. Key Management Server

The NTS-KM server primarily uses the security features provided by TLSv1.3. These include authenticity, message integrity and confidentiality during data transmission. Moreover, the protocol offers protection against common attacks such as spoofing, packet manipulation and replay. Since the NTS Key Establishment protocol only allows a TLSv1.3 handshake or higher, downgrade attacks to TLSv1.2 or lower can be ruled out. However, note that a DoS attack on the key management server can prevent new connections or parameter updates for secure PTP communication. A hijacked key management server is also critical, because it can completely disable the protection mechanism. A redundant implementation of the key management server is therefore essential for a robust system. A further mitigation can be the limitation of the number of TLS requests of single PTP nodes to prevent flooding. These rules depend on the specific PTP use case.

C. Secured PTP Communication

Special requirements apply for secured time transmission in general, which are described in RFC 7384 [24]. The protection of PTP messages by an AuthTLV in combination with the NTS-KM approach provides authenticity, message integrity and key freshness. It also provides protection against external Man-in-the-Middle attacks such as spoofing, packet manipulation, replay attack and rogue master attack. No encryption of the messages takes place, as this is not seen necessary for time information. Based on the approach of Immediate Security Processing, common group keys are used, which limits the source authenticity. Hence, everyone in the group is able to change the messages. A hijacked PTP node or the break of the group key is therefore the greatest threat.

This approach offers no protection against grandmaster time source attacks as well as packet interception and removal. For this reason, additional security mechanisms (Prong C and Prong D) need to be considered.

VI. Key Management for PTP – Details

This chapter extends Section IV and delves into the details of the GroupRequest and GroupResponse messages. This section also describes the necessary configurations on the PTP node and NTS-KE server side.

A. Requirements and Pre-Configuration

Before secure PTP communication is feasible, both the PTP nodes and the KM server must be set up. Therefore, both sides must be equipped with a private key and a certificate in advance. The certificate contains a digital signature of the CA as well as the public key of the sender. The key pair is required to establish an authenticated and encrypted channel for the initial TLS phase. Distribution and update of the certificates can be done manually or automatically. However, it is important that they are issued by a trusted CA instance, which can be either local (private CA) or external (public CA).

The configuration of the PTP nodes is limited to specifying the Port and IP address (or host name) of the KM server and the group assignment. This can be linked to both the PTP domain and the sdoId (optional). The KM server can also define groups, limit the number of members, or specifically exclude individual participants. One possible way of forming groups is to use a X.509 certificate in which group names are used as part of the *Subject* name. In addition, the KM server defines how often the security parameters (SP and SA) will be updated and which PTP messages are to be protected with which SA.

B. Phase 1: NTS Key Establishment Protocol

The following sections describe the structure of the messages defined here and the function of the individual parameters derived from the PTPv2.1 specification.

1) Stage 1: TLSv1.3-Handshake: The communication starts at the PTP node by establishing a TLSv1.3 connection to the key management server. Both parties authenticate each other based

on the certificates and the KM server checks whether the PTP node is authorized to access the secure PTP network. According to the NTS specification, the ALPN extension "ntske/1" [7, ch. 4] must be used to signal the following NTS Key Establishment protocol.

2) Stage 2: The GroupRequest Message: After a successful handshake, the PTP node sends a GroupRequest message over the encrypted TLS channel. This is a sequence of *Records*, which have a TLV-like format (see Figure 3). Records contain a Critical Bit (C) that rules the handling of unknown Types [7, ch. 4].

A GroupRequest message contains four Records (see Figure 4). The message starts with the *NTS Next Protocol Negotiation* [7, ch. 4.1.2], which signals the use of "PTPv2.1" as the following time protocol. The Records *SdoId Request* and *PTP Domain Request* allow the PTP node to make a request to join a desired group. The sdoId is the composition of the *majorSdoId* and the *minorSdoId* [3]. Then the *End of Message* Record signals the end of the GroupRequest.

3) Stage 3: The GroupResponse Message: After receiving the GroupRequest, the key management server first checks the *NTS Next Protocol Negotiation* Record, which must contain the identifier for "PTPv2.1". Then a GroupResponse message with a dynamic number of Records is constructed (see Figure 5).

The message also starts with the *NTS Next Protocol Negotiation* that confirms "PTPv2.1". This is followed by the *Group Access Indicator* which grants or denies the access to the desired group. If the access is prohibited, only an *End of Message* follows. The *Security Policy* determines which PTP message types in the confirmed group are to be secured with which *Security Association*. The Security Association contains information on how the PTP messages are to be secured and comprises all the parameters required to build or check an AuthTLV. If the SP specifies that the PTP message types are to be secured differently, several SA Records are included. The server generates the SPs and SAs upfront in cyclical intervals and stores them in its databases (SPD, SAD). SAs that have expired or are no longer used by any PTP peer can thus be deleted. Before we will explain the Records *Security Parameter Validity Period* as well as *Security Parameter Update Time,* the content of an SA will be detailed. Its composition follows a fixed structure (see Figure 6), which is based on the parameters specified in [3, ch. 16.14.2] and is supplemented by a KeyID. Due to the immediate-verification approach described here, some fields contain fixed values.

C	Record Type	Body Length
Record Body		

Fig. 3. Structure of an NTS Record

NTS Next Protocol Negotiation
SdoId Request
PTP Domain Request
End of Message

Fig. 4. A *GroupRequest* message: sequence of four Records

NTS Next Protocol Negotiation
Group Access Indicator
Security Policy
Security Association
[Security Association]
Security Parameter Validity Period
Security Parameter Update Time
Next Group Access Indicator
Next Security Policy
Next Security Association
[Next Security Association]
Next Security Parameter Validity Period
Next Security Parameter Update Time
End of Message

Fig. 5. A *GroupResponse* message: sequence of multiple Records (green fields only available during the Update Time period)

Thus, *immediateSecurity* must always take the value TRUE, *sequenceNoLength* and *resLength* the value zero. The Security Parameter Pointer (*SPP*) and *keyId* are unique identifiers and are controlled by the key management server. Other parameters such as *seqenceID window*, *KeyLength*, *IntegrityAlgTyp* and *icvLength* are defined by the administrator in the KM server as part of the security policies. *KeyLength* and *icvLength* are also dependent on the *IntegrityAlgTyp*. HMAC-SHA256-128 is used as the *IntegrityAlgTyp*, since it must be supported by all PTP nodes. The MAC algorithm is stored as an Object Identifier (OID) with dot notation in *IntegrityAlgTyp*. In order to be able to parse this field, the additional field *IntegrityAlgTypLength* was added, which specifies the variable number of *I* octets in *IntegrityAlgTyp*. The symmetric *key* is a cryptographically secure random number and is used for the ICV calculation.

Another Record in the GroupResponse message is the *Security Parameter Validity Period*, which specifies the validity period of all parameters (SP and SA) contained in the message. This includes the time data *NotBefore* and *NotAfter* in a Unix timestamp format. Another Record is *Security Parameter Update Time*, in which the KM server specifies the point in time at which new security parameters are available. This time must lie within the current validity period. Both records together allow a cyclical update of the Security Policy, the Security Association and the group constellation. The update interval is determined by the key management server and can be configured by the administrator.

Field	Octets	Offset
SPP	1	0
immediateSecurity	1	1
sequenceNoLength	2	2
resLength	2	4
sequenceIdWindow	2	6
integrityAlgTypeLength (*I bytes*)	2	8
integrityAlgType	I	10
icvLength	2	10 + I
keyId	4	12 + I
keyLength (*K bytes*)	2	16 + I
key	K	18 + I

Fig. 6. Structure of the *Security Association* Record body

When the beginning of the time in *Security Parameter Update Time* is reached, all GroupResponse messages add further Records, which contain the new parameters for the following validity period. To simplify matters, these have been marked with the prefix "Next" (green fields in Figure 5) and are structurally identical to the "current" Records. If a group member is excluded in the following period, new security parameters are not included in the response and the *Next Group Access Indicator* Record contains a *deny*. More details about the updating process will be described below in chapter VI.D.1.

After sending the GroupResponse message, the server closes the TLS channel correctly by sending a *close notify* to the PTP node. The PTP nodes store the parameters in the local SP and SA databases after receiving and checking the GroupResponse message. Now the PTP node is able to build the AUTHENTICATION TLVs for the respective PTP group as well as to generate and check the ICVs. The negotiation of the parameters is now complete.

C. Phase 2: Securing PTP Messages

In this phase the PTP nodes are able to verify or create secured messages within an assigned PTP group. The procedure for protecting and checking PTP messages is described as follows.

1) Protection of PTP Messages: The securing process of the message to be sent starts with the interpretation of its Policy Limiting Fields (see Figure 7). These are in particular: *sourcePortIdentity*, *domainNumber*, *majorSdoId*, *minorSdoId* as well as *messageType* and are already present in the header of the PTP message. Depending on the PTP implementation, this information can also be provided elsewhere. The PLF are then used to determine the associated SP in the Security Policy Database (SPD), since a device with multiple ports can also have multiple SPs. Based on the security policy the decision is made whether the message type is to be secured. If so, the SP contains a Security Parameter Pointer (SPP), which can then be used to determine the corresponding SA in the Security Association Database (SAD). The information contained in the SA enables the PTP node to construct the AUTHENTICATION TLV [3, ch. 16.14.3]. Due to the Immediate Security Processing used here, most fields contain fixed values (see Figure 8).

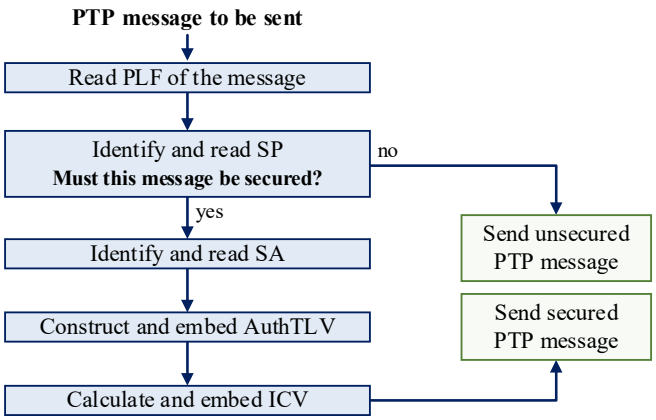

Fig. 7. PTP securing process

Field	Octets	Offset	Value
tlvType	2	0	0x8009
lengthField	2	2	0x16
SPP	1	4	<var>
secParameterIndicator	1	5	0x0
keyID	4	6	<var>
[disclosed Key]	D	10	0x0
[sequenceNo]	S	10+D	0x0
[RES]	R	10+D+S	0x0
ICV	K	10+D+S+R	<var>

Fig. 8. Structure of an AUTHENTICATION TLV with unused fields (gray)

The *tlvType* must be of the type "AUTHENTICATION" and has the value 0x8009 [3, ch. 14.1.1]. The *lengthField* specifies the payload length of the TLV excluding *tlvType* and *lengthField*. Since no optional fields are included in Immediate Security Processing and HMAC-SHA256-128 is used in this approach, the value of the *lengthField* is always 22 octets (1+1+4+16). The SPP corresponds to the value from the SP and enables the receiver to assign the associated SA. The *secParamIndicator* is a tuple of Boolean values and signals the presence of optional fields. Since these are not required, this is equivalent to the value 0. The *keyID* identifies the applied key of the used SA. Since this approach defines only one key per SA, the value is constant over the SA lifetime. Finally, the *ICV* is the checksum calculated over the complete PTP message, which depends on the HMAC algorithm and the key used. With the insertion of the AuthTLV at the end of the PTP message including the ICV, the message is considered secured and can be transmitted.

2) Verification of PTP Messages: To verify a PTP message, the recipient must also read the PLF and thus identify the SP in his SPD (see Figure 9). If the received message is secured according to the SP, it is checked whether an AuthTLV is present at the end of the PTP message. Subsequently the SPP is read from it and the corresponding SA is determined from the SAD.

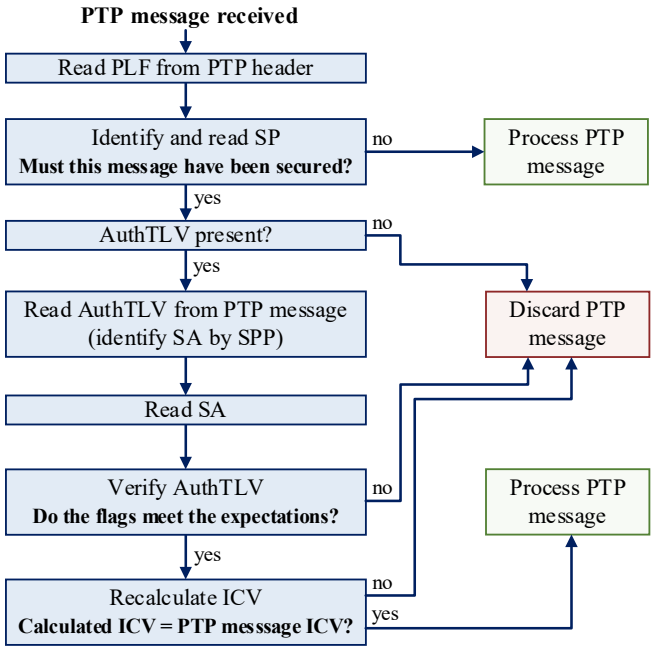

Fig. 9. Message verification process

Then the PTP node checks whether the SequenceID in the PTP header meets the expectations and lies within the SequenceID window defined in the SA to prevent packet replay. If the remaining parameters in the AuthTLV match the data in the SA (e.g. various flags), the integrity check is initiated. Together with the parameters from the SA, the PTP node generates an ICV over the PTP message and compares it to the ICV in the AuthTLV. If both are identical, the authenticity of the message is confirmed and can be further processed by the PTP stack.

D. Features and Limitations

This section explains the functionality and properties of security parameter rotation and considers the challenges in unicast connections.

1) Rotation of the Security Parameters: The rotation mechanism is an important part of this approach. It enables both the updating of security policies and security parameters (e.g. the key), as well as an easy way of group control. Thus, this method ensures key freshness without interruption of a running and secured PTP communication. The two parameters *Validity Period* and *Update Time*, which are transmitted as part of a GroupResponse, form the core of this mechanism. To prevent a potential attack on this method, a loosely time synchronous NTS key management server is mandatory. The time synchronization of the NTS-KM server should be both redundant and secure (e.g. with NTS-secured NTP).

Figure 10 shows a possible example where a six-hour rotation has been set by the administrator on the NTS-KM server. Of course any other key exchange frequency is conceivable as well. The validity period of the parameters received in the GroupResponse starts at 0:00 and ends at 6:00 of the current day (left blue period). These points in time are stored as Unix timestamps and are shown here in simplified form. If this time range refers to the currently used security parameters, the current time of the PTP node must also lie within this area. In this example, the update time parameter is set to 5:00. PTP nodes that are already in the secured PTP network send a GroupRequest at a random time after the update time (5:00) has been reached and before the current parameters have expired (6:00) in order to obtain the new parameters. This prevents load peaks on the key management server caused by simultaneous requests from the PTP nodes upon expiration of the parameters. This mechanism also ensures that all PTP nodes have already received the new parameters when the current parameters expire and can continue operating without interruption (right green period). Expired parameters are then deleted after a defined bridging time (e.g. 10 seconds). The bridging time ensures that PTP packets that were generated and sent during the rotation can still be checked.

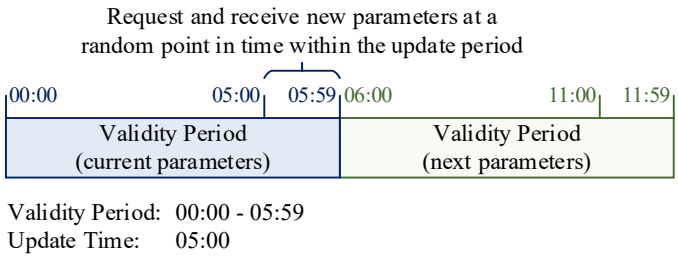

Fig. 10. Example of a security parameter rotation process (the green box shows the Validity Period of the upcoming parameters)

The simple approach also has its limitations. An immediate group adjustment (e.g. the exclusion of a PTP node) or update of the security parameters is currently not possible. By establishing the TLS connection to query the parameters, the resulting traffic linearly (O(n)) increases in relation to the number of secured PTP nodes. But due to the moderate size of typical PTP networks and the low request frequency of the PTP nodes to the NTS-KM server, the load is insignificant. An extension with the function of an immediate update requires another connection between PTP node and KM server. Since in this scenario the data transfer originates from the KM server, a PTP node would have to listen permanently on a defined UDP port or a multicast address. One solution would be comparable to GDOI and the GROUPKEY-PUSH method [10, ch. 4] defined in it.

2) Challenges in Unicast Communication: Since the automatic key management system presented above has been coupled with immediate security processing and is therefore group-based, it is particularly suitable for multicast connections. Using of the KM system for securing unicast connections is possible in principle, but involves further challenges. To create such a unicast connection, a group with only two participants (master and slave) may be formed. However, there are restrictions if Transparent Clocks lie within this path. These must also be added to the "Unicast" group if support for correction fields is required. It becomes more difficult if there are other PTP devices (e.g. Boundary Clocks) between Grand master and slave. The security and authenticity of the time information thus strongly depends on the topology of the PTP network and its participants. If one pursues complete end-to-end security, this may not be solvable in any case with the procedure introduced here. One possible approach in those situations is the TESLA protocol, which can be used as a supplement, but increases the complexity considerably (see also Section III.B).

There are basically two important challenges with unicast connections: source authentication and scalability. High source authenticity is achievable by establishing an exclusive and secure connection between master and slave. Apart from these two PTP nodes, the key required for this should only be known to the NTS-KM that establishes this connection. The support of Transparent Clocks, which also require the key for this purpose, remains problematic. A possible approach here is topology recognition by the NTS-KM, which determines the paths between slave and master and determines the TCs.

To achieve good scalability, the unicast approach requires dynamic grouping (for TC support) or pairing between two PTP nodes. A reasonable method for key distribution is a ticket system. The very simplified process could look like this: A master registers once with the NTS-KM via TLS and receives a symmetric key $K_{M\text{-}KE}$ that only both of them know. A slave that wishes a unicast connection to the master informs the NTS-KM via a *UnicastRequest* over the TLS channel and receives a *UnicastResponse* including a ticket in addition to the unique master-slave key $K_{M\text{-}S}$. This ticket is encrypted with $K_{M\text{-}KE}$ and contains the $K_{M\text{-}S}$ as well as other security parameters. The slave can now generate a PTP message (e.g. *Sync_Request*), which is secured with $K_{M\text{-}S}$ and contains the embedded ticket. The master is able to decrypt this ticket with $K_{M\text{-}KE}$ and subsequently verify the PTP message with $K_{M\text{-}S}$. Of course, the master is also able to generate secured PTP messages with this key. The advantage of

this approach is that the tickets do not require a TLS connection from the KM server (but only from the PTP nodes) to distribute the keys. Obviously, other aspects like replay protection and topology changes in the PTP network have to be considered in this proposal. The detailed elaboration of the NTS-based key distribution for PTP unicast connection is still necessary and therefore not part of this paper.

VII. Proof-of-Concept Implementation

The following section describes the characteristics and first results of a PoC-implementation [25]. This is followed by a brief examination as well as a performance test of the HMAC and CMAC algorithm.

1) Protocol Implementation and Results: As part of the concept development, we started the creation of a first minimal implementation of the key management system. Currently, the PoC-implementation developed for Linux allows the authentication and negotiation of parameters between a PTP node and the KM server. The software is written in C++17 and also uses the *OpenSSL 1.1.1g* and *Asio C++* libraries to provide the TLS functionality and other cryptographic operations. Initial testing confirms the basic functionality of the negotiation and the automatic update of security parameters. Actually, the PoC implementation is not linked to a PTP implementation and therefore still simulates the PTP instances. An integration of the PoC in the previous work from [19] (Linux PTP) or [20] (PTPd) would be useful and enable further practice-oriented analyses. While the size of a GroupRequest is constant at 21 octets, the data amount of the GroupResponse varies depending on the security policies defined. If the GroupResponse only contains one SA at a time, the size of this message is 115 octets. If it also contains the parameters for the following validity period, the message grows to 215 octets. When each PTP message type is protected by its own SA, then 10 SAs are sent for a validity period. Therefore, the GroupRequest can reach a maximum size of 1223 octets. According to initial measurements, the generation time of a typical GroupResponse (with one SA) on a desktop PC (Intel i9-9900K) takes around 0.3µs, but strongly depends on the implementation and hardware performance and must be investigated further.

2) Comparison of HMAC and CMAC: The high accuracy in PTP is especially due to the exact capture of the transmit and receive timestamps. PTP defines the transmit timestamp as the transmission start of the first symbol after the Start of Frame Delimiter (SFD) of an Ethernet frame. The transmission of this timestamp by PTP can be done in one-step or two-step mode, depending on the configuration and hardware support. In two-step mode, a Follow-up message transfers the transmit timestamp and is therefore not time critical. In one-step mode, on the other hand, this timestamp is written into the PTP message while the sending process of this message is already taking place. Depending on the transmission speed of the network interface, only a few microseconds remain for this operation. It becomes even more difficult if the PTP packet is to be additionally secured by an ICV. The ability to do this depends primarily on the amount of data to be secured, the MAC algorithm and the hardware performance.

A secured PTP packet typically consists of a PTP header, a PTP body and an AuthTLV. The header has a constant length of 34 octets, the body 10 to 30 octets, depending on the message type, and the AuthTLV constant 26 octets (including 16 octets for the ICV). Therefore, 54 to 74 octets per PTP packet (ICV excluded) must be secured by the MAC algorithm. The maximum time allowed for the securing process also depends on the total length of the data to be sent. If PTP uses pure Ethernet (802.3) as the network protocol (the PTP packet is embedded as an Ethernet payload), there are an additional 14 octets of the Ethernet header between the SFD and the ICV. This results in a total length between SFD and ICV of 68 to 88 octets. In order to support the one-step mode, the timestamp must be written to the PTP packet during the transmission of these data quantities and the ICV must be calculated and embedded. At an Ethernet transmission speed of 100 Mbit/s, this results in 5.44µs to 7.04µs. If PTP uses IPv4 and UDP as network protocols instead, an additional 28 octets (IPv4 header: 20 octets; UDP header: 8 octets) are transmitted. For this total length of 96 to 116 octets, this leads to a transmission time of 7.68µs to 9.28µs. However, this problem becomes even more critical at higher transmission speeds (1 GBit/s or higher).

The PTPv2.1 standard recommends the use of HMAC-SHA256-128 [3, ch. 16.14.3.9] for the securing process. However, an analysis in [5, ch. IV] showed that this algorithm is not optimal due to the long block size of 512 bits and that AES-128-CBC-CMAC is a much better alternative. Thus, a performance test was executed on two different devices, in which an increasing amount of random data was secured by a MAC algorithm. The allocation and freeing of memory was excluded in these measurements. Figure 11 shows the results on a Meinberg microSync RX. Even without hardware-supported AES acceleration, CMAC (fine-staged green line) is superior to

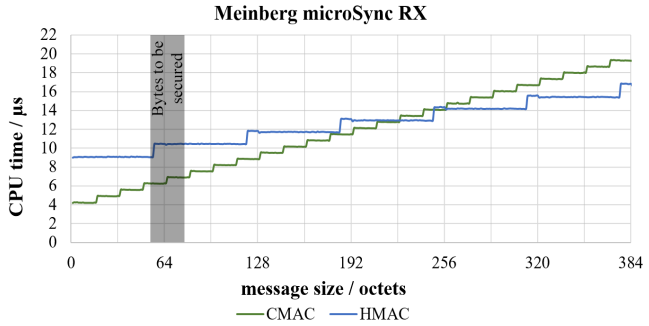

Fig. 11. Comparison of computing time for CMAC and HMAC (microSync)

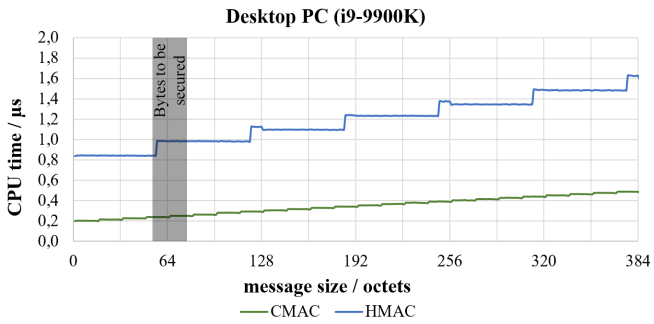

Fig. 12. Comparison of computing time for CMAC and HMAC (desktop PC)

the HMAC (course-staged blue line) algorithm. The visible gradations result from the block size of the respective algorithm. On the microSync, CMAC needs 6.22µs for 54 octets and HMAC 9.04µs. For 74 octets accordingly 6.87µs and 10.46µs. As an additional comparison, Figure 12 shows the process on a desktop PC with an Intel i9-9900K processor and active AES acceleration. For both measurements, the library OpenSSL 1.1.1g was used as well. The CMAC (lower green line) algorithm shows significantly better results than HMAC (upper blue line) and should therefore be included in a future version of the PTP standard. A purely software-based solution (without AES acceleration) cannot meet the timing requirements. A hardware-specialized implementation of the MAC algorithms (e.g. in FPGAs) can further reduce the latency and thus enable one-step mode in PTP even with transmission speeds of 1 Gbit/s and above.

VIII. Conclusion And Further Work

The concept presented and the results of the PoC implementation illustrate the easy realization of an automatic key management system for PTPv2.1 based on the NTS key exchange. In conjunction with the PTP integrated security mechanism (Prong A), this allows immediate security processing of PTP messages. Moreover, the NTS-KM offers the additional use of NTS-secured NTP, which can be used to increase the redundancy of the PTP network. The examination of the securing duration of PTP messages on industrial hardware also showed that CMAC is more suitable than HMAC, which is especially important in one-step mode.

Further work is concentrating on the optimization and expandability of this concept by additional functions. This can be e.g. an instant parameter update method in case of a detected attack or the usage of *Key Tables* in each SA, to support multiple keys per rotation period. Another aspect is the addition of the *Delayed Security Processing* or the usage of secured unicast connections, by extending this approach with a *UnicastRequest* and *UnicastResponse*. The focus here lies on providing strong source authentication in combination with Immediate Security Processing.

References

[1] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," IEEE Std 1588-2002, Oct. 2002, doi: 10.1109/IEEESTD.2002.94144.

[2] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002), July 2008.

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

[4] N. Moreira, J. Lázaro, J. Jimenez, M. Idirin and A. Astarloa, "Security mechanisms to protect IEEE 1588 synchronization: State of the art and trends," 2015 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS), Beijing, 2015.

[5] C. Önal and H. Kirrmann, "Security improvements for IEEE 1588 Annex K: Implementation and comparison of authentication codes," 2012 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication Proceedings (ISPCS), San Francisco, CA, USA, Sep. 2012.

[6] A. Treytl and B. Hirschler, "Security flaws and workarounds for IEEE 1588 (transparent) clocks," 2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS), Brescia, Italy, Oct. 2009.

[7] 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.

[8] 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.

[9] M. Langer, K. Teichel, K. Heine, D. Sibold and R. Bermbach, "Guards and Watchdogs in One-Way Synchronization with Delay-Related Authentication Mechanisms," 2019 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS), Portland, OR, USA, Sep. 2019.

[10] B. Weis, S. Rowles and T. Hardjono, "The Group Domain of Interpretation," RFC 6407, doi 10.17487/rfc6407, Oct 2011.

[11] A. Perrig, D. Song, R. Canetti, J. D. Tygar and B. Briscoe, "Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction," RFC 4082, June 2005.

[12] "IEEE Standard for local and metropolitan area networks-Media Access Control (MAC) Security," in IEEE Std 802.1AE-2018 (Revision of IEEE Std 802.1AE-2006), Dec. 2018.

[13] S. Kent and K. Seo, "Security Architecture for the Internet Protocol," RFC 4301, doi 10.17487/rfc4301, Dec. 2005.

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

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

[16] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, doi 10.17487/rfc8446, Aug. 2018.

[17] D. Sibold, S. Roettger and K. Teichel, "Network Time Security," Internet Draft, draft-ietf-ntp-network-time-security-08, March 2015.

[18] P. Kemparaj and S. S. Kumar, "Secure precision time protocol in packet switched networks," 2019 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS), Portland, OR, USA, Sep. 2019.

[19] E. Shereen, F. Bitard, G. Dán, T. Sel and S. Fries, "Next Steps in Security for Time Synchronization: Experiences from implementing IEEE 1588 v2.1," 2019 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS), Portland, OR, USA, Sep. 2019.

[20] D. Maftei, R. Bartos, B. Noseworthy and T. Carlin, "Implementing Proposed IEEE 1588 Integrated Security Mechanism," 2018 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS), Geneva, 2018.

[21] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)," RFC 7296, Oct. 2014.

[22] K. Teichel and G. Hildermeier, "Experimental evaluation of attacks on TESLA-secured time synchronization protocols," in Security Standardisation Research, C. Cremers and A. Lehmann, Eds. Springer International Publishing, 2018, pp. 37-55.

[23] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen and T. Kivinen, "Internet Security Association and Key Management Protocol (ISAKMP)," RFC 2408, doi 10.17487/rfc2408, 1998.

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

[25] K. Heine and M. Langer, "NTS-KM PoC implementation," GitLab Repository, [Online] available: https://gitlab.com/kai_heine/ntske4ptp, 2020.