Next Steps in Security for Time Synchronization: Experiences from implementing IEEE 1588 v2.1

Ezzeldin Shereen[1], Florian Bitard[1], György Dán[1], Tolga Sel[2], and Steffen Fries[2]

[1]Division of Network and Systems Engineering, EECS, KTH Royal Institute of Technology, Sweden
[2]Siemens AG, Corporate Technology, Germany

***Abstract*—The lack of integrated support for security has been a major shortcoming of Precision Time Protocol version 2 (PTPv2) for a long time. The upcoming PTPv2.1 aims at addressing this shortcoming in a variety of ways, including the introduction of lightweight message authentication. In this paper we provide an overview of the planned security features, and report results based on an implementation of the proposed integrated security mechanism based on the open source Linux PTP, including support for hardware timestamping. Our implementation includes an extension of Linux PTP to support transparent clocks. We provide results from an experimental testbed including a transparent clock, which illustrate that the extensions can be implemented in software at a low computational overhead, while supporting hardware timestamping. We also provide a discussion of the remaining vulnerabilities of PTP time synchronization, propose countermeasures, and discuss options for key management, which is not covered by the standard.**

I. Introduction

A long-standing weakness of PTPv2 has been the lack of support for security controls. As PTP is starting to find adoption in critical infrastructures, e.g., in power systems as an alternative for GPS-based synchronization of Phasor Measurement Units (PMUs), concerns about the potential consequences of a compromise of PTPv2 protocol messages have become more severe. Recent work has shown that time synchronization attacks against PMUs could bypass existing security controls with potentially severe consequences [1, 2]. While recent works on the detection of time synchronization attacks are able to detect low rate attacks with high accuracy [3], defense in depth calls for standards support for PTP protocol message authentication.

Experimental support for protocol message authentication for PTP was introduced first in Annex K of PTPv2 [4], but several works pointed out that the overhead of the proposed message authentication scheme was too high. Subsequent work proposed in PTPv2.1 (IEEE 1588 v2.1) [5], which is currently being standardized in the IEEE, redefined the security scheme with various options to accommodate protocol internal security means for authentication, and protocol agnostic means provided by the embedding infrastructure. The PTP protocol internal option targets different authentication options, which allow adaptation to the target deployment environment.

In this paper we provide an implementation-based feasibility analysis of the proposed protocol internal security features in the upcoming PTPv2.1 standard on top of the open-source Linux PTP implementation of PTP. Compared to previous studies as [6], our implementation is based on a different open-source software, includes a transparent clock, supports hardware timestamping on Layer 2 transport, delayed unauthenticated processing, and combined immediate and delayed processing. We also discuss remaining vulnerabilities in time synchronization, propose potential countermeasures, and we highlight potential avenues for future development and standardization.

The rest of the paper is organized as follows. Section II describes the security features introduced in PTPv2.1. In Section III we present the security extensions we implemented to Linux PTP. In Section IV we demonstrate the effect of our security extension on the synchronization accuracy and the computation overhead using a testbed implementation. Section V discusses the remaining vulnerabilities and open issues. Section VI concludes the paper.

II. Security in PTPv2.1

PTPv2.1 (IEEE 1588 v2.1) is currently in the last phase of specification in IEEE and is expected to become a standard in 2019. In contrast to the predecessor IEEE 1588 v2 describing security in an experimental Annex K, IEEE 1588 v2.1 goes one step further and defines a security option integrated into the Precision Time Protocol directly as well as additional measures leveraging existing or defining new security means by the embedding infrastructure to support security.

A. Overview

IEEE 1588 v2.1 addresses security in various ways described as prongs and depicted in Figure 1. The following provides an overview about the prongs.

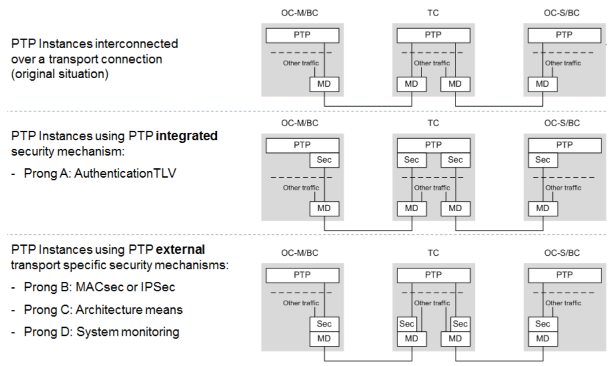

OC-M/BC
TC
OC-S/BC
PTP Instances interconnected over a transport connection (original situation)
PTP
Other traffic
MD
PTP Instances using PTP **integrated** security mechanism:
- Prong A: AuthenticationTLV
Sec
PTP Instances using PTP **external** transport specific security mechanisms:
- Prong B: MACsec or IPSec
- Prong C: Architecture means
- Prong D: System monitoring

Fig. 1: IEEE 1588 v2.1 Security Prongs.

Prong A - PTP Integrated Security Mechanism is described as an option in IEEE 1588 v2.1 section 16.14. Prong A specifies an Authentication TLV integrated into and aligned with the PTP message structure to specifically support environments not featuring security protocols, which also protect PTP (see Prong B below). This Authentication TLV is designed to support different modes of operation, namely immediate or delayed security processing of the message authentication information. Immediate processing allows for instant verification of integrity, while delayed authentication performs the authentication at a later point in time, when the security parameters for the verification are available. To achieve this, different types of key management are necessary. The security provided with the Authentication TLV is always relying on a symmetric key. Note that based on the security policy, the delayed authentication can be divided into two approaches; (1) delayed unauthenticated, in which the message processing is performed, and in case the delayed authentication fails, the action is rolled back, and (2) delayed authenticated, in which the execution is postponed until the delayed authentication has been performed. The requirement of either rolling back an action or postponing the execution until authentication was performed provides the boundary conditions for delayed authentication in general. Applicability depends on the actual target use case. Note that the Authentication TLV may be utilized multiple times in the same packet. This also allows the combination of immediate and delayed authentication modes. For the management of the key material and the associated security policy, two different automated key management schemes are outlined in Annex S of IEEE 1588 v2.1. Immediate security processing can be supported by utilizing the group-based key management approach GDOI [7]. Delayed security processing is supported by the multicast authentication scheme TESLA [8]. Note that IEEE 1588 v2.1 does not require the support of these key management approaches and also allows manual key management.

Prong B – PTP External Transport Security Mechanisms is part of the informative Annex S and describes the application of security means to protect PTP message transport, which may be already in place. Specifically addressed is the protection of layer 2 traffic as hop-to-hop protection using IEEE 802.1x MACsec or using IPsec to provide a secure tunnel between entities.

Prong C – Architecture Guidance, which is also part of the informative Annex S provides recommendations for architectural enhancements in the PTP architecture like redundant communication paths and/or Grand Master Clocks (GM) to detect attacks and improve resilience.

Prong D – Monitoring and Management Guidance is also part of the informative Annex S, and gives additional recommendations for the embedding infrastructure, which support the detection of potential attacks (e.g., DoS attacks not countered by cryptography). Beyond these are the detection of tampering and degradation (delays) and/or failures of network equipment and media, the utilization of additional performance monitoring tools being specified by the Management subcommittee, as well as guidelines for security network management interfaces. Specifically the latter can also be connected to Prong B, if IEEE 802.1x is used for link layer authentication and authorization.

More detailed information about the security approach can also be found in [9]. Nevertheless, as the focus here is the implementation of the integrated security option, which has been updated in IEEE 1588 v2.1 in the meantime, more insight is provided to the current definition in the following subsection.

B. *Prong A Authentication TLV*

The Authentication TLV in the integrated security option has been changed slightly since the publication of [9] as a result of reviews and discussions within the IEEE. We only present the final status here, as it was the base for the implementation. Figure 2 shows the final definition.

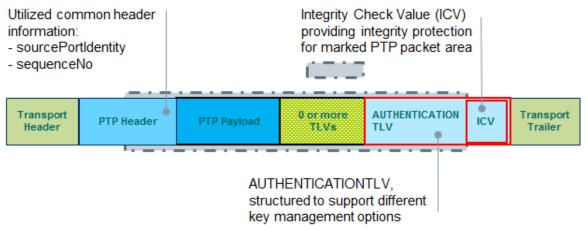

Utilized common header information:
- sourcePortIdentity
- sequenceNo
Integrity Check Value (ICV) providing integrity protection for marked PTP packet area
Transport Header
PTP Header
PTP Payload
0 or more TLVs
AUTHENTICATION TLV
ICV
Transport Trailer
AUTHENTICATIONTLV, structured to support different key management options

Fig. 2: IEEE 1588 v2.1 Protected PTP Packet

In contrast to the previous version, the security indication in the PTP header has been deprecated as it was seen as not contributing to the overall security. In contrast, as it is assumed that security is enforced by a policy, a wrong statement in the security indication could have been used as an attack vector. The Authentication TLV definition itself, shown in Figure 3, has not been changed.

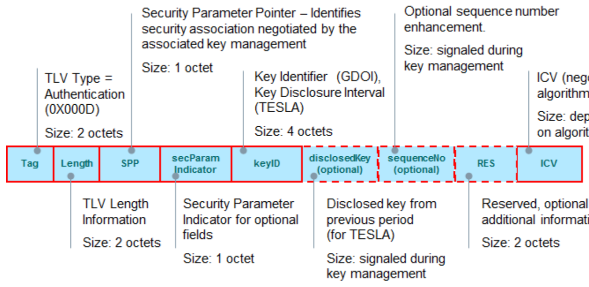
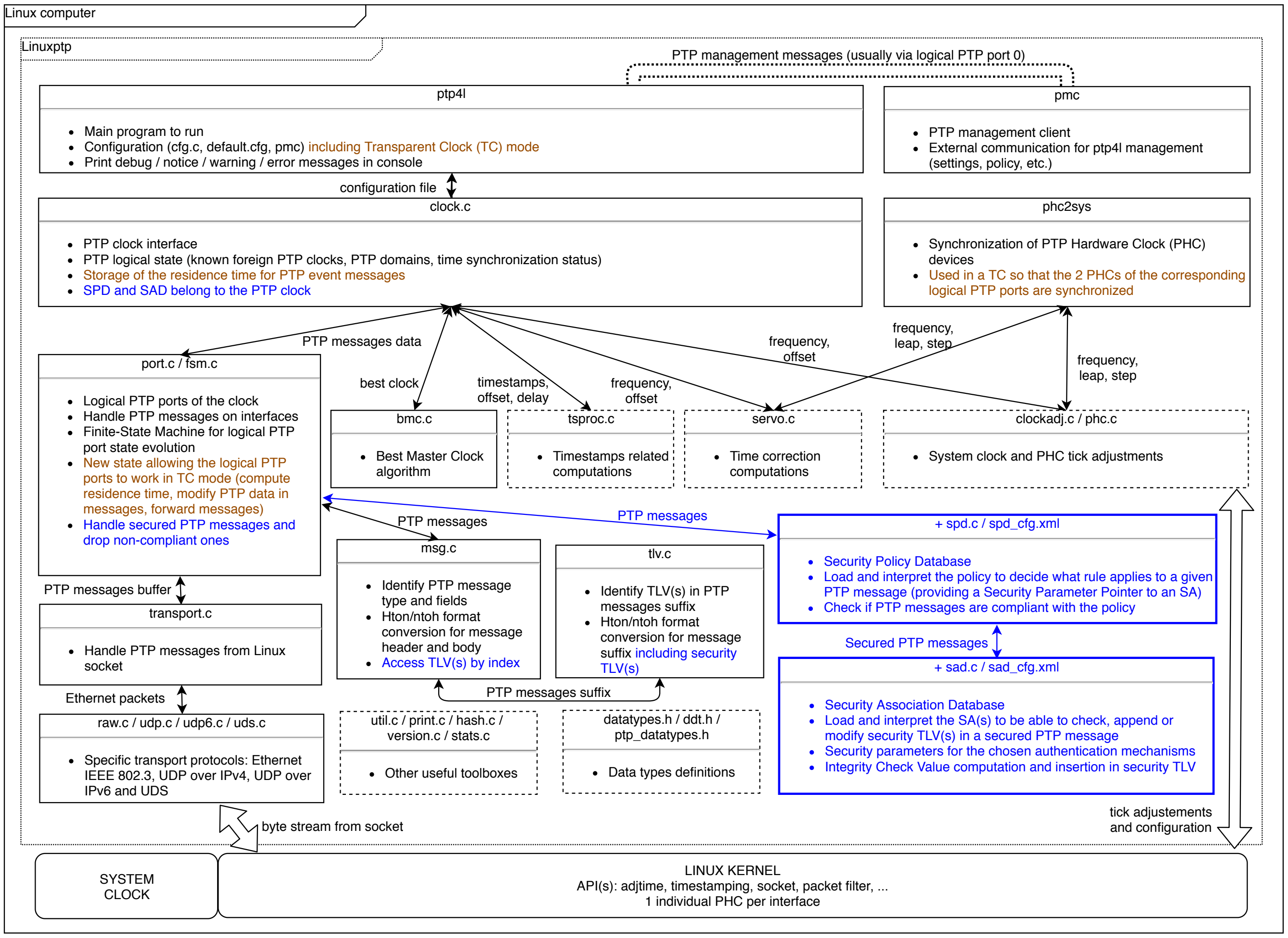

Security Parameter Pointer – Identifies security association negotiated by the associated key management
Size: 1 octet
Optional sequence number enhancement.
Size: signaled during key management
TLV Type = Authentication (0X000D)
Size: 2 octets
Key Identifier (GDOI), Key Disclosure Interval (TESLA)
Size: 4 octets
ICV (negotiated algorithm)
Size: depending on algorithm
Tag
Length
SPP
secParam Indicator
keyID
disclosedKey (optional)
sequenceNo (optional)
RES
ICV
TLV Length Information
Size: 2 octets
Security Parameter Indicator for optional fields
Size: 1 octet
Disclosed key from previous period (for TESLA)
Size: signaled during key management
Reserved, optional additional information
Size: 2 octets

Fig. 3: IEEE 1588 v2.1 Authentication TLV

III. Implementation

We implemented the integrated security mechanism as defined in Prong A by extending v1.8 of Linux PTP. In order to be able to test the security mechanism in a complete configuration, our implementation included the extension of Linux PTP v1.8 to support acting as a transparent clock. In terms of the implementation and validation of the security mechanisms, our focus was on the power profile, i.e., we implemented all the functionality required for peer-to-peer timing on top of Layer 2 transport (Ethernet), with support for hardware timestamping.



Linux computer
Linuxptp
PTP management messages (usually via logical PTP port 0)
ptp4l
Main program to run
Configuration (cfg.c, default.cfg, pmc) including Transparent Clock (TC) mode
Print debug / notice / warning / error messages in console
pmc
PTP management client
External communication for ptp4l management (settings, policy, etc.)
configuration file
clock.c
PTP clock interface
PTP logical state (known foreign PTP clocks, PTP domains, time synchronization status)
Storage of the residence time for PTP event messages
SPD and SAD belong to the PTP clock
phc2sys
Synchronization of PTP Hardware Clock (PHC) devices
Used in a TC so that the 2 PHCs of the corresponding logical PTP ports are synchronized
PTP messages data
best clock
timestamps, offset, delay
frequency, offset
frequency, offset
frequency, leap, step
frequency, leap, step
port.c / fsm.c
Logical PTP ports of the clock
Handle PTP messages on interfaces
Finite-State Machine for logical PTP port state evolution
New state allowing the logical PTP ports to work in TC mode (compute residence time, modify PTP data in messages, forward messages)
Handle secured PTP messages and drop non-compliant ones
bmc.c
Best Master Clock algorithm
tsproc.c
Timestamps related computations
servo.c
Time correction computations
clockadj.c / phc.c
System clock and PHC tick adjustments
PTP messages
PTP messages
+ spd.c / spd_cfg.xml
Security Policy Database
Load and interpret the policy to decide what rule applies to a given PTP message (providing a Security Parameter Pointer to an SA)
Check if PTP messages are compliant with the policy
msg.c
Identify PTP message type and fields
Hton/ntoh format conversion for message header and body
Access TLV(s) by index
tlv.c
Identify TLV(s) in PTP messages suffix
Hton/ntoh format conversion for message suffix including security TLV(s)
PTP messages buffer
transport.c
Handle PTP messages from Linux socket
Secured PTP messages
+ sad.c / sad_cfg.xml
Security Association Database
Load and interpret the SA(s) to be able to check, append or modify security TLV(s) in a secured PTP message
Security parameters for the chosen authentication mechanisms
Integrity Check Value computation and insertion in security TLV
PTP messages suffix
Ethernet packets
raw.c / udp.c / udp6.c / uds.c
Specific transport protocols: Ethernet IEEE 802.3, UDP over IPv4, UDP over IPv6 and UDS
util.c / print.c / hash.c / version.c / stats.c
Other useful toolboxes
datatypes.h / ddt.h / ptp_datatypes.h
Data types definitions
byte stream from socket
tick adjustements and configuration
SYSTEM CLOCK
LINUX KERNEL
API(s): adjtime, timestamping, socket, packet filter, ...
1 individual PHC per interface

Fig. 4: Linux PTP architecture, including extensions for transparent clock (brown) and for security processing (blue).

A. *Transparent clock extension*

Linux PTP is implemented as a finite state machine (FSM), in which the state transitions depend on the received messages, timers and the clock type, which can be ordinary, boundary, management, or through our extension, transparent. In order to implement the transparent clock, we thus modified the corresponding transitions of the FSM. In addition to these modifications, we added support for forwarding *Announce*, *Sync*, *Follow_Up*, *PTP management message* and *Signaling message* messages, including the appropriate modification of the *correction* field based on the residence time of the messages in the transparent clock. We also implemented functionality to support the creation of *Follow_Up* and *Pdelay_Resp_Followup* messages, so as to support two-step mode, which was needed for implementing security processing without hardware support.

In order to support message forwarding we configured Linux to act as a bridge, using *iproute2*. In addition, since the quad-port network interface cards we utilized in our testbed (Intel i340-T4) use a separate clock for each physical interface, we decided to synchronize the clocks of the different ports via the Linux PTP Hardware Clock (PHC) subsystem. In principle, lack of synchronization between the different ports should not have a major impact on synchronization accuracy as long as the the residence times of messages are in the order of microseconds (as the clock drift between ports is negligible), but we decided to synchronize the clocks so as to avoid any unnecessary noise.

Figure 4 shows a block diagram of Linux PTP, including the modifications made for the implementation of the transparent clock functionality, marked with brown color.

B. *Security processing*

We implemented two of the three modes of authentication discussed in the upcoming IEEE 1588 v2.1 standard: immediate processing and delayed unauthenticated processing. In addition, our implementation supports the combined use of these two authentication modes. We also consider the possibility that a transparent clock may have different security policies on different ports, and hence the authentication modes and the security associations might have to be changed when forwarding protocol messages (*Announce*, *Sync*, *Follow_Up*, *PTP management message* and *Signaling message*).

For incoming messages, Linux PTP performs network to host byte order (N2HBO) conversion *in situ* in the message buffer. In order to implement security processing, we thus have to create a copy of the incoming message before N2HBO

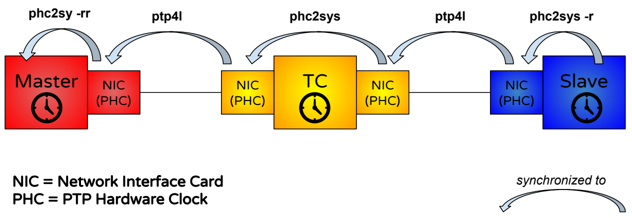


Fig. 5: Illustration of the experimental testbed, consisting of a master clock, a transparent clock, and a slave clock.

conversion is done. After the N2HBO conversion we then process the PTP header, we look up the security policy in the Security Policy Database (SPD), and check whether the *AUTHENTICATION TLV* headers correspond to the policy. If yes, we process the *AUTHENTICATION TLV*, starting from the last one present in the message. In the case of an *AUTHENTICATION TLV* specifying immediate processing, we use the previously stored copy of the message for the *Integrity Check Value (ICV)* calculation. For delayed unauthenticated processing in the slave, we add the copy of the message to a linked list and perform authentication upon key disclosure. In case of a failed ICV check we add an error message to the system log. We do not perform delayed unauthenticated processing in the transparent clock. We note that the variable structure of the *AUTHENTICATION TLV* increases the complexity of processing incoming messages significantly.

For creating the *AUTHENTICATION TLV* for outgoing messages we add the *AUTHENTICATION TLV* header(s) before host to network byte order (H2NBO) conversion is done, we then compute the ICV value after H2NBO conversion, and set it in the *AUTHENTICATION TLV* after performing H2NBO on the ICV. For creating the *AUTHENTICATION TLV* for forwarded messages we compare the security policy of the outgoing message to that of the incoming message. If delayed unauthenticated processing is used then we copy the corresponding *AUTHENTICATION TLV* from the incoming message, since the transparent clock does not perform verification. In the case of immediate processing we do not copy the incoming *AUTHENTICATION TLV*, but we create a new *AUTHENTICATION TLV* based on the SPD, the Security Association Database (SAD), and the message content. Doing so facilitates supporting different security policies on different ports. Figure 4 shows the modifications made for the implementation of security processing marked with blue color.

C. Key management

We implemented manual key management in the form of two modules, one for the SPD and one for the SAD. Both databases are stored in XML format, specifying the authentication mode and the security association per source port, destination port and message type. The SAD for delayed unauthenticated mode contains the value of the head (trust anchor) of the hash chain for the master, and the tail (actual value) for the slave, together with the hash algorithm, the length of the hash chain, and the disclosure interval. The transparent clock does not verify messages with delayed authentication, and thus it only checks if the *AUTHENTICATION TLV* is present as mandated by the SPD.

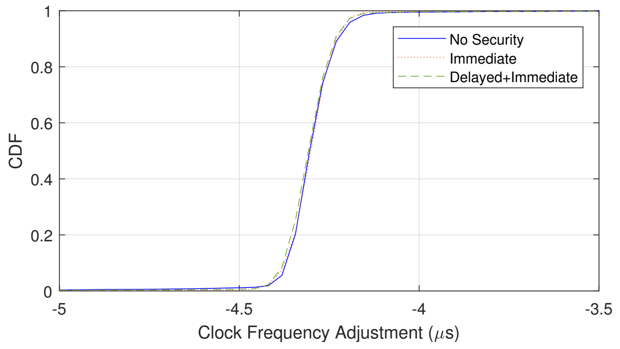


Fig. 6: CDF of the implemented frequency adjustments without security, with immediate processing and with combined delayed-immediate processing.

IV. Experimental evaluation

In what follows we show results from our implementation of the *AUTHENTICATION TLV* in two-step mode. We created a testbed that consists of a Master Clock, a transparent clock, and a slave clock. Each clock runs on an HP ProLiant ML G9 server equipped with an Intel i340-T4 quad port NIC card capable of hardware timestamping, on Ubuntu 16.04.1 LTS. Figure 5 shows the testbed components and their connections. In the experiments we considered three scenarios:

1) No Security: all security features are disabled.
2) Immediate: immediate security processing for all messages.
3) Mixed (Delayed + Immediate): immediate processing for peer delay messages, delayed unauthenticated processing for *Announce* and *Sync* messages, and both mechanisms for *Follow_Up* messages.

Synchronization Accuracy: Figure 6 shows the cumulative distribution function (CDF) of the frequency adjustments computed by the clock servo of the slave clock for the three scenarios. The adjustments were collected over 10 interleaved intervals of 30 minutes each, in order to suppress the impact of natural clock frequency variations. The figure shows that the distribution of the adjustments is not affected by security processing. To confirm this, we performed a two-sample Kolmogorov-Smirnov test pairwise on the empirical distributions, and the data passed the test at a confidence level $\alpha = 0.05$, which indicates that the data for the three scenarios come from the same probability distribution. Based on these results, we can conclude that the security extension has no effect on the synchronization accuracy.

Security Processing Delays: Figure 7 shows the CDF of the residence time (time between receiving and forwarding the message) in the transparent clock for *Sync* messages sent by the Master Clock. The figure shows that the use of immediate processing mode can significantly increase the residence time, which is expected since the ICV is verified for every received message and is recomputed for every message before it is

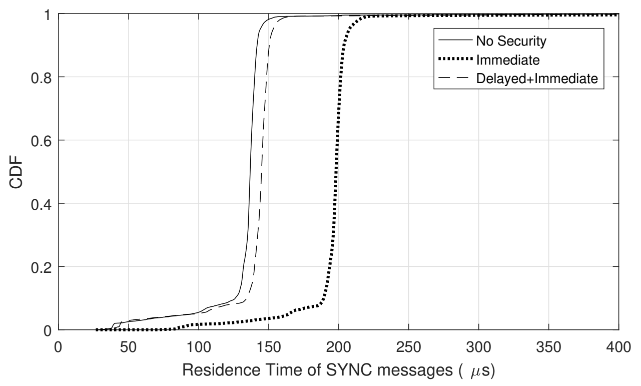


Fig. 7: CDF of the residence time of Sync messages in the Transparent clock without security, with immediate processing and with combined delayed-immediate processing.

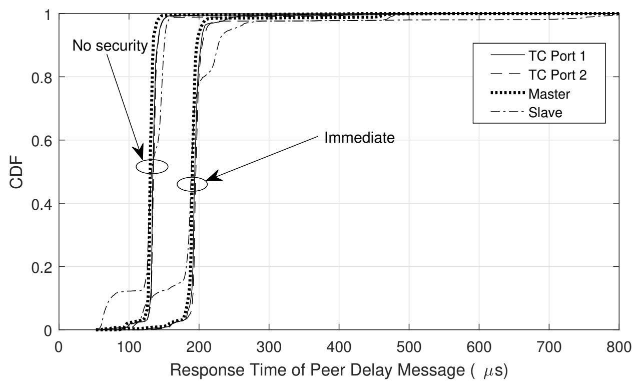


Fig. 8: CDF of the response time for peer delay messages without security and with immediate processing, when sent by the TC, the Master and the Slave in the testbed.

forwarded. We observe on average $70\mu s$ extra delay introduced by immediate processing. At the same time, the mixed use of immediate and delayed authentication processing introduced relatively little extra delay (on average $10\mu s$, compared to No Security), as the ICV is only computed for a subset of the messages. It is to be noted that although the messages are further delayed in the transparent clock, the synchronization performance is not affected as the increased residence time is accounted for by the correction field. We observe similar behavior in Figure 8 for the peer delay messages, used to estimate the delay between neighbouring clocks. The figure shows the CDF of the time between receiving a *Pdelay_Req* message and sending a *Pdelay_Resp* message measured at the NICs of the following devices: (1) port 1 of the transparent clock, (2) port 2 of the transparent clock, (3) the Master Clock, and (4) the slave clock. For all the ports, we observe an additional delay of $70\mu s$ on average, similar to the delay experienced by *Sync* messages (c.f., Fig. 7).

CPU Usage: To measure the CPU usage of Linux PTP at the slave clock we disabled 15 of the 16 threads of the Intel Xeon E5-2620V4 CPU, and reduced the CPU frequencey to its minimum (1.2 GHz). We then used *pidstat* to measure the average CPU usage over one minute intervals. Without security we measured a user space CPU usage of 8ms per minute. With immediate security processing the corresponding time was 24ms, and it was 27ms in the mixed scenario. The three fold increase is mostly due to the ICV computation, and related overhead, and should be accounted for in dimensioning the computing power of embedded devices. Security processing did not affect the CPU usage of the kernel, as expected, as the system calls are mainly related to message transmission and reception using the Linux socket API.

V. Remaining Vulnerabilities and Challenges

The proposed *AUTHENTICATION TLV* is clearly a significant step forward in securing PTP time synchronization. Nonetheless, there are several remaining threats and issues that should be considered when planning deployments in adversarial environments.

A. Hardware Implementation Challenges

Besides the complexity of managing the variable length *AUTHENTICATION TLV* in hardware, it is unclear how integrated security processing would affect timing accuracy in one-step mode. One potential solution for accurate hardware transmit timestamping in one-step mode would be to use constant time execution cryptographic functions for computing the *ICV*, so that the transmit timestamp of messages can be known before sending. An alternative would be to use cryptographic functions with bounded worst case execution time combined with a hard-real time scheduler on the NIC, which allows tight control of the message transmission time.

B. Known Vulnerabilities

By its design, delayed processing does not allow to verify the integrity of the correction field, which is updated in every transparent clock upon forwarding *Follow_Up* messages. Thus, the integrity of the correction field can only be verified using immediate processing.

Group key compromise: Immediate processing relies, however, on symmetric key cryptography through the application of a group key. It is thus vulnerable to the compromise of any of the PTP nodes using the same security association. To mitigate this threat, the number of devices using the same security association should be minimal. One possibility would be to use a different security association between every pair of PTP nodes. While seemingly not scalable, typical physical network topologies are rather sparse, and hence the number of edges (i.e., keys) is likely to be in the order of the number of nodes. Our implementation of the *AUTHENTICATION TLV* shows that it is feasible to use different security associations on different ports of the same node without major overhead. As a complement, key material could be protected by using a secure cryptoprocessor, such as a Trusted Platform Module (TPM), for storing the symmetric keys and for ICV calculation.

Software compromise: The above countermeasures do not mitigate the threat of a compromised transparent clock. Since immediate processing does not protect the correction field while within the clock, an attacker could manipulate it unnoticed, so as to create asymmetric one way delays. This threat could be mitigated by storing and processing all data related to PTP in a trusted execution environment, such as Intel SGX or ARM TrustZone, but we are not aware of implementations supporting this, yet. We thus argue that it is important to target transparent clocks in the security assessment for the intended operational environment to derive the applicable security requirements and based on this the appropriate measures. This is important also when a certain security level according to IEC 62443-3-3 is to be met in industrial automation systems.

Delay attack: The cryptography-based security control defined in Prong A does not counter delay attacks. Thus, a composition of the different Prongs defined in IEEE 1588 v2.1 is necessary to provide appropriate protection.

C. Key Management and Profiles

From a standard evolution perspective, IEEE 1588 v2.1 will provide protocol integrated security options as well as external security measures. But it will also leave room for further specification. In particular, the application of the two named key management approaches, GDOI and TESLA, needs further specification to allow for interoperable implementations.

GDOI as defined may not be directly applicable out of the box to IEEE 1588 v2.1. It was defined targeting IPsec and therefore the defined payloads match what is needed for IPsec. Investigations to what payload definitions are necessary to transport the parameters for protecting PTPv2.1 are required. As GDOI has been enhanced with key data payloads to support application in the power system automation domain as IETF RFC 8052 [10], analysis is necessary, if the defined key data payloads may be directly reused. This would lower the effort for specification and potentially also for implementation effort.

TESLA on the other hand may be applied for the setup of the hash chain and the key calculation. Nevertheless, for applying TESLA it is also required to distribute further information like the anchor value, the key disclosure interval and further settings. There exists already an IETF standard RFC 4442 [11] defining the bootstrapping of TESLA utilizing a different group supporting key management protocol called MIKEY (Multimedia Internet Keying). To not mix too many key management protocols, the approach described in RFC 4442 may be directly transferred to GDOI allowing it to provide both security parameter information for immediate processing, and security parameter provisioning for TESLA to enable delayed security processing of PTP messages.

In addition, as several other domain specific standards profiling the options provided by IEEE 1588, a security profile may be necessary. This could be a generic security profile as an extension to the upcoming IEEE 1588 v2.1 combining different security options and key management choices into one profile. Alternatively, there already exist domain specific standards profiling PTP for different applications for different verticals. Examples are

- PTP Power System Application Profile (defined in IEEE C37.238 and IEC 61850-9-3)
- PTP Industry Profile (IEC 62439-3)
- PTP Telecom Profile for Phase/Time (ITU-TG.8265.1).

As these profiles are already defined, they may need to be enhanced to profile the security options of IEEE 1588 v2.1 for their specific application domain.

VI. Conclusion

We presented an experimental evaluation of the *AUTHENTICATION TLV* to be introduced in PTPv2.1. Our implementation shows that the proposed extension can be implemented in software with a relatively low computational overhead, while making use of hardware timestamping, thus without affecting the achievable synchronization accuracy. In addition, we discussed remaining threats and potential work items for standardization to pave the road for adopting PTP and PTP security in different application domains.

References

[1] S. Barreto, M. Pignati, G. Dán, J. L. Boudec, and M. Paolone, “Undetectable timing-attack on linear state-estimation by using rank-1 approximation,” *IEEE Trans. on Smart Grid*, vol. 9, pp. 3530–3542, Jul. 2018.

[2] S. Barreto, J. L. Boudec, E. Shereen, G. Dán, M. Pignati, and M. Paolone, “A continuum of undetectable timing-attacks on PMU-based linear state-estimation,” in *Proc. of IEEE SmartGridComm*, Oct. 2017.

[3] E. Shereen and G. Dán, “Correlation-based detection of PMU time synchronization attacks,” in *Proc. of IEEE SmartGridComm*, Oct. 2018.

[4] IEEE Instrumentation and Measurement Society. TC-9 Sensor Technology, “Standard for a precision clock synchronization protocol for networked measurement and control systems,” IEEE 1588-2008, 2008.

[5] IEEE Instrumentation and Measurement Society. TC-9 Sensor Technology, “Draft Standard for a precision clock synchronization protocol for networked measurement and control systems,” IEEE P1588 v2.1 Draft D1.4, 2018.

[6] D. Maftei, R. Bartos, B. Noseworthy, and T. Carlin, “Implementing proposed ieee 1588 integrated security mechanism,” in *Proc. of IEEE ISPCS*, Sep. 2018.

[7] B. Weis, S. Rowles, and T. Hardjono, “The Group Domain of Interpretation,” RFC 6407, IETF, Oct. 2011.

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

[9] K. O’Donoghue, S. Fries, and D. Sibold, “New security mechanisms for network time synchronization protocols,” in *Proc. of IEEE ISPCS*, Sep 2017.

[10] B. Weis, M. Seewald, and H. Falk, “Group Domain of Interpretation (GDOI) Protocol Support for IEC 62351 Security Services,” RFC 8052, IETF, June 2017.

[11] S. Fries and H. Tschofenig, “Bootstrapping Timed Efficient Stream Loss-Tolerant Authentication (TESLA),” RFC 4442, IETF, June 2006.