This full text paper was peer-reviewed at the direction of IEEE Instrumentation and Measurement Society prior to the acceptance and publication.

A Security Analysis and Revised Security Extension for the Precision Time Protocol

Eyal Itkin
Tel Aviv University, Israel
eyalitki@post.tau.ac.il

Avishai Wool
Tel Aviv University, Israel
yash@eng.tau.ac.il

***Abstract*—The Precision Time Protocol (PTP) aims to provide highly accurate and synchronized clocks. Its defining standard, IEEE 1588, has a security section ("Annex K") which relies on symmetric-key secrecy. In this paper we present a detailed threat analysis of the PTP standard, in which we highlight the security properties that should be addressed by any security extension. During this analysis we identify a sequence of new attacks and non-cryptographic network-based defenses that mitigate them. We then suggest to replace Annex K's symmetric cryptography by an efficient elliptic-curve Public-Key signatures. We implemented all our attacks to demonstrate their effectiveness, and also implemented and evaluated both the network and cryptographic defenses. Our results show that the proposed schemes are extremely practical, and much more secure than previous suggestions.**

I. Introduction

A. PTP Overview

IEEE 1588 is a standard defining the Precision Time Protocol (PTP), [10]. An initial version of the standard was published at 2002, and a second version was later published in 2008. PTP is aimed specifically at measurement, control and financial applications, applications that have an increasing need for highly accurate and synchronized clocks.

The PTP standard defines a distributed network of clocks, called *PTP nodes*, that dynamically builds a master-slave hierarchy, which collectively achieves the desired accuracy. In addition, PTP is bidirectional, allowing each node to calculate the unique network delay between itself and the central clock, called the *Grand Master*.

One of the main novelties of PTP is the fact that the design has provisions for the use of hardware timestamping, as explained in [27]. From the need to reach very high accuracy and precision, rose the need to bypass the jitter, or noise, induced by the message passing through the network stack in the OS. This is done using PTP-aware network interface cards (NICs), that are able to timestamp the message in the lowest hardware levels, just before the message is sent to the underlying physical layer.

B. Related Work

In 2011, Mizrahi [14] suggested to make use of the known IPSec solution, and to base the security Type-Length-Value (TLV) extension header on it. Later on, in 2012, after the results of [17], Kirrmann suggested a modified Annex K. His main changes were to remove the redundant 3 way-handshake, to update the Message Authentication Code (MAC) that is used in the Integrity Check Value (ICV) calculation and to use IPSec for the key distribution scheme. A proposal by Ellegaard [6], speaks about hop-by-hop security based on the MACsec. These proposals speak only about the means to establish a secure transport channel between the 2 PTP nodes, regardless of the nodes' role in the protocol, or the protocol's behavior.

A proposal by Fries [8], speaks mainly on the key distribution scheme, and suggests to use a standard cryptographic algorithms for the ICV calculation. Sibold/Dickerson [24] [5] suggest using TESLA [19] for the Security Association (SA) and to check the applications of Network Time Security (NTS), a standard that is currently underway. While NTS presents a new asymmetric approach, master X509 certificates, the main use of it is for the bootstrapping of TESLA.

As we can see, the papers mainly address the key distribution problem, that is not part of the original Annex K, or suggest new ways to establish a secure channel between the PTP nodes. We argue that one of the main design flaws in Annex K is its inability to handle an insider threat, as will be shown in the next sections, and the related works do not address this issue.

Nevertheless, in [15] there is a suggestion for a new methodology, based on a "divide and conquer" technique. In an internal paper of the security subcommittee, [21], they present a "four pronged approach" which suggests 4 level of security. We chose to follow the "four pronged approach", and present a fully functioning *Prong A*, End-to-End strong source authentication, security extension for IEEE 1588.

C. Contributions

In this paper we present a detailed threat analysis of the PTP standard, in which we highlight the security properties that should be addressed by any security extension. We identified a sequence of new attacks that can be mounted against PTP, depending on the attacker's strength: from out-of-band network attacker to corrupt insiders. We then identified network defenses that mitigate some of the attacks: binding the clock-IDs to the network addresses, and expanded use of message sequence numbers to prevent spoofing and introduce session-like semantics into the protocol. While the basic version of these defenses is fully compatible with the PTP message format, we suggest a small modification - to use 2 reserved

978-1-4673-9615-8/16/$31.00 ©2016 IEEE

bytes and extend the sequence ID to 32 bits - which greatly improves the protocol's resilience to attack.

We then suggest to replace Annex K's symmetric cryptography by an efficient elliptic-curve Public-Key signature scheme. The proposed EdDSA signatures are only 64 bytes long, and only need to be attached to some of the grandmaster messages. We implemented all our attacks to demonstrate their effectiveness, and also implemented and evaluated both the network and cryptographic defenses. Our results show that the proposed schemes are extremely practical, and much more secure than previous suggestions.

Organization In the next section we introduce the main aspects of PTP. Section 3 presents the attack scenarios and the threat analysis. In Section 4 there are details of our proposed security extension. We then show experimental results of our defense mechanism, and compare it to the state of the art security proposals. We conclude with Section 5. Full details can be found in our technical report [9].

II. Preliminaries

The PTP protocol consists of two main layers, and an additional management layer:

1) Grandmaster election - the Best Master Clock (BMC) algorithm
2) Time oriented messages - Sending timestamps and measuring network delay

The first layer is a distributed leader election algorithm that is being constantly calculated by all of the master-candidate PTP nodes. Each candidate node sends an `ANNOUNCE` message in which it declares its management priority, its time source, etc., and the best clock is elected as the "grandmaster" clock. The leader election algorithm is called "Best Master Clock" (BMC) algorithm. The elected master clock then multicasts its accurate timestamp to all of the slave nodes, using the 2nd layer of the protocol.

The 2nd layer of the protocol consists of the following messages, as shown in Figure 1:

1) `SYNC` - The master's timestamp, broadcast to all PTP nodes every #seconds
2) `DELAY_REQ` - A call from a slave to measure the delay with the master
3) `DELAY_RESP` - An answer from the master with its receive timestamp

A. 2-Step sync

Hardware timestamping mechanisms break support for most cryptographic primitives since encryption or signatures applied by a higher level in the software stack must happen **before** the message is sent. Therefore, the standard defines a notion of *2-step* communication, in which a PTP-aware NIC recognizes the `SYNC` message, calculates t_1 and stores it locally. Then the software layers query the NIC, obtain t_1, and embed its value in the body of the `FOLLOW_UP` message (see Figure 1), which enables cryptographic operations to be done on the `FOLLOW_UP` message - including the timestamp.

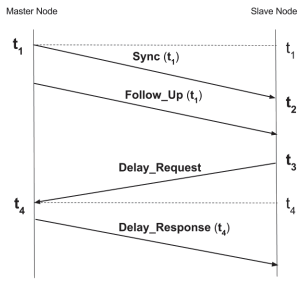

Fig. 1. PTP 2-step message flow. Note the 4 timestamps t_1, t_2, t_3, t_4: the `FOLLOW_UP` and `DELAY_RESP` messages carry t_1 and t_4 to the slave.

B. PTP Annex K

IEEE standard 1588 defines an experimental security extension in its Annex K. The goals of the security protocol are: providing group source authentication, message integrity, and prevention of replay attacks.

The Annex K security protocol is based on a preshared symmetric key: when a node sends messages to an untrusted destination node, it first performs a 3-way handshake in which both sides authenticate to each other using the preshared key. Every message sent over to a given Security Association (SA) has an authentication Type-Length-Value (TLV) extension header, that contains an Integrity Check Value (ICV) that is calculated over all of the message, including anti-replay counters.

C. Testing Environment

In our work we used *DeterProject*'s infrastructure, [2] and [11], in order to experiment and test our results. The experimentation was done on the open source linux PTP daemon, *PTPd* [22]. We implemented our proposed security extension on top of the daemon's 2.3.1.1 version.

III. Attack Scenarios and Mitigations

The Precision Time Protocol consists of 3 main protocol layers:

1) Event messages - Time oriented Messages
2) Best Master Clock Algorithm - grandmaster election
3) Management Layer

In the following subsections we introduce the 3 main arms races between the protocol's adversaries, fully described in [9], and the proposed defense mitigations. Each arms race corresponds to a different protocol layer, thus covering all of the protocol's main layers.

A. Event Messages Attacks

In this subsection we focus on the attack scenarios that target the time synchronization of the PTP nodes. These types of attacks have received the majority of security-aware attention so far, as in [26]. In the following arms race we discuss our suggested defense mitigations, and show the experimental results of their deployment.

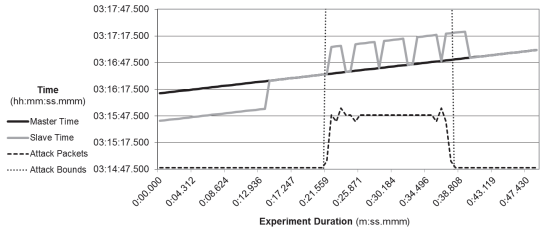

Fig. 2. Applicative Sync Spoofing: Between times 0:21 and 0:37 the adversary is sending hostile SYNC messages, with timestamps of +30 seconds.

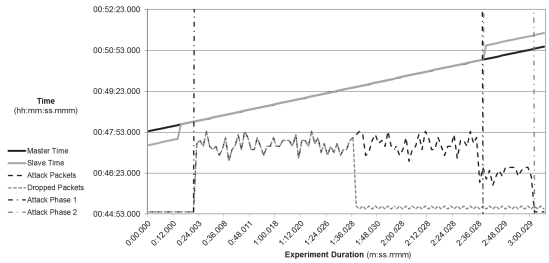

Fig. 3. Phase I: performing a blind window snatching attack on window of size 50. Sending packets at a rate of 10 pps, the attack goes through 65536/50 ($\approx$ 1300) packets in slightly more than 2 minutes. Phase II: Sync Spoofing.

1) Applicative Sync Spoofing Attack Scenario:

a) Attack: Send spoofed SYNC messages with hostile timestamps on behalf of the grandmaster, to a chosen target node. The targeted node will adopt these hostile timestamps and will recalibrate itself accordingly. This attack can be performed by a very weak network adversary.

Figure 2 shows what happens when the attacker sends bogus SYNC messages; the slave accepts the false timestamps and sets its clock to the attacker's chosen time. Once every 5 sec the legitimate grandmaster's messages reset the slave back to the true time. Note that the +30 sec value is illustrative; we successfully set the clock 20 years back.

b) Mitigations: The major design flaw that enables the *Sync Spoofing* attack is the fact that there is no binding between the PTP entity (the clock ID) and the underlying network ID. Such a binding would enable the slave node to overcome the attacks using the following checks:

1) Sending messages to the master's network address as derived from its clock ID
2) Checking for a match between the PTP Clock ID and the network ID and discarding mismatching messages

To do so we suggest to construct the IDs based on the underlying network IDs instead of using IEEE EUI-64 assigned clock IDs, see details in [9].

2) Network Sync Spoofing Attack Scenario:

a) Attack: A slightly stronger attacker can still deploy the *Sync Spoofing* attack: it needs only to spoof the underlying network addresses too.

b) Mitigations: The adversary is making use of the fact that the messages from the master to the slave are treated without session semantics. Although the PTP header defines the sequenceId field, it's only used for distinguishing the messages from one another.

We suggest to extend the standard in the following way:

1) Master originating messages will start with a pseudo-random sequence ID counter, each message type will have a separate counter
2) Each counter will be incremented by 1 for each sent message
3) The slave node will validate a received message's sequence ID, and check that it matches a predefined range of available ids. The range of counter values acceptable to the slave is called the *Window*.

In addition, the suggested extension can be used to strengthen the delay/response mechanism of the protocol:

4) Each REQUEST_<X> message will have a pseudo-random sequence ID, acting as a challenge

In [9] we show that this mitigation indeed blocks both flavors of the *Sync Spoofing* attack.

3) Blind Window Snatching Attack Scenario:

a) Attack: The attacker can use two ideas to give him an advantage: (a) he can transmit much faster than the real grandmaster, and (b) he can use the defender's Window against him. Note that after a valid message is received there is no need for the slave to wait for the "old" messages, and the window can be advanced.

Using these ideas, a network adversary can deploy a *blind window snatching* attack. In this attack the adversary scans the sequenceId range ($R = [0, 65535]$) and sends a message matching the maximal ID in each window. Once the attacker "hits" the slave's window, he will "snatch" it and advance it fast in each sent message. After this first phase, the target's window will wrap-around to 0 and the second phase of the attack can be performed.

As can be seen in Figure 3, once the adversary "catches" the target's session window (at time 1:36), all the master's messages fall outside the window, resulting in logged message drops.

The message complexity of the attack on a target with sequenceId from range R, a window size of w, and an original attack with message complexity of K is:

$$\#messages = \frac{|R|}{w} + K \tag{1}$$

b) Mitigation: The blind window snatching attack depends on the rather small 16-bit range R caused by the 2 byte size of sequenceId. We suggest to overcome this limitation by enlarging the sequenceId field using 2 of the reserved bytes. An updated header structure and the result of the mitigation, can be found in [9].

4) Inband Sync Spoofing Scenario:

a) Attack: An insider adversary can also perform the *Sync Spoofing* attack described in Section III-A2. The difference is that the adversary's position enables him to sniff the

session "secrets" (sequence numbers), therefore the previous defenses are ineffective.

b) Mitigation: Because the adversary is In-Band, the only way to withhold secrets from him is by using a cryptographic solution. While Annex K suggests using a symmetric key approach, we suggest to utilize a Public-Key cryptographic solution, that can defend against much stronger (Insider) attacks as well.

5) PTP Insider Threat:

a) Attack: Assuming Annex K is activated with its symmetric-key defenses, a hostile PTP node can pretend to be the master node and initiate the 3-way handshake against the target slave. By using the legitimate symmetric key, the handshake will complete successfully resulting in the slave being fooled to think that the adversary is the legitimate master node. From now on, the adversary will use the derived symmetric key to send hostile `SYNC` messages to his target.

b) Mitigation: This attack scenario addresses a major flaw in the standard's Annex K: there is no way to differentiate between the slave nodes and the master nodes. This leads to an inherent vulnerability when addressing an insider threat. Variants of the described attack will work against any other non-role based defense scheme deployed in the network, including the use of standard solutions such as IPsec or MACsec as was suggested in [17].

We suggest a new approach, a role-based solution, that will prevent a slave node from masquerading as the master node. The proposed solution makes use of s Public-Key based cryptography, and is described in full details in Section IV.

B. BMC Attacks

The BMC-chosen grandmaster clock is responsible for the synchronization of the PTP slave nodes, thus making the BMC algorithm a desired target for an adversary. In this section we focus on attack scenarios that target the BMC algorithm itself.

1) Insider Rogue Master Attack Scenario:

a) Attack: A hostile PTP node will propose himself as a grandmaster candidate by sending fake `ANNOUNCE` messages declaring him to be the best clock in the network. This can easily be done by faking to be a truly magnificent clock: Using the minimal value for the `Priority1` and `Priority2` fields; declaring that the clock's source as `ATOMIC_CLOCK`; etc. Annex K, and any symmetric key solution, will fall short of defending against such a threat. This is due to the fact that the insiders have the secret keys and can appear as a convincing grandsmaster.

We implemented and tested this attack. Figure 4 shows the attack's effects—note that even after the adversary stops his attack (at time 0:44), the network continues according to his hostile time, since the original grandmaster continues to dictate the adversary's hostile timestamp to the network.

b) Mitigation: The solution is the same as for the PTP insider threat (III-A5): using public key cryptography.

C. Management Layer Attacks

The IEEE 1588 standard [10] defines a PTP specific management format. The format enables the management node

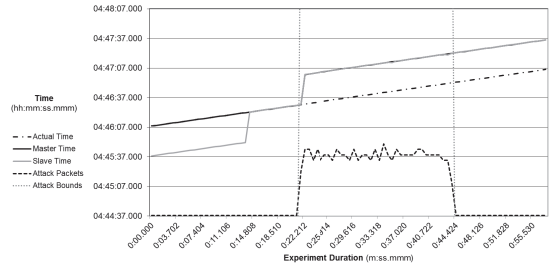

Fig. 4. Rogue Master: An outsider adversary sends `ANNOUNCE` messages to nominate himself to be the grandmaster.

to query and update the PTP nodes' dataset fields, using `GET_<X>` and `SET_<X>` messages. Although the protocol defines extensive management capabilities, it does not specify any authentication mechanism.

1) Proxy Grandmaster Attack Scenario:

a) Attack: Even the weakest applicative adversary can send fake management messages to upgrade a specified PTP node's dataset. This dataset improvement will cause the targeted node to win the BMC leader election, thus declaring him the network's grandmaster. Once elected, the adversary will send `SET_TIME` messages to directly control the target's time, thus taking full control of the time of the whole PTP network.

b) Mitigation: We agree with [18] and recommend to deprecate the protocol. The management messages can easily be replaced by a standard SNMP ([25]) implementation that maintains the Set and Get basic operations and natively supports authentication.

IV. Proposed Security Extension

In this section we describe the details of our cryptographic defense mechanism, its analysis compared to Annex K and state of the arts security proposals, and its remaining gaps.

A. Prerequisites and Scope

Our defense scheme assumes the existence of several prerequisites:

1) The existence of a management entity
2) A predefined public verification key, distributed to all PTP nodes
3) Master certificates signed by the management entity

In this paper we focus solely on the PTP header format and its basic set of messages, our solution can easily be extended to the rest of the messages. In the described solution the clock IDs are built in accordance to the underlying network addresses.

B. Cryptographic Design Choices

We have implemented and demonstrated our cryptographic solutions. We chose an Edwards-Curve (Ed) based public key scheme: the EdDSA signing scheme [20], a choice that is based on 4 main arguments:

1) Only the authenticity is important
2) Ed schemes make use of relatively short keys
3) Ed schemes make use of relatively short signatures
4) Ed schemes are designed to be faster than other digital signatures schemes

In our implementation over the open-sourced PTPd, we integrated an Ed25519 scheme [3], based on the *WolfCrypt* library [28]. Using this scheme the size of a signature is 64 bytes, and the size of the public key is 32 bytes, thus maintaining the rather small size of the original messages.

C. PTP Header Format

Since easing the deployment process was one of our design goals, we made every effort to keep the protocol's original header format. The only extensions we suggest to the header format were already discussed in Section III-A3b:

1) Using 2 reserved bytes for the enlarged `sequenceId`
2) Using 1 reserved bit, from the bit flag, marking the use of the defense scheme

While the enlarged `sequenceId` enables session-like semantics between the master and the slave, it also acts as a cryptographic aid: an effective anti-replay counter. By making the slave check the window validity of this field, and by incrementing it slowly through the large 32 bit range, the adversary's ability to replay a recorded signed message is practically non-existent.

The second use of the enlarged `sequenceId` field is as a challenge-response field. Before the slave sends a `DELAY_REQUEST` message, it randomly picks a `sequenceId`, therefore generating a challenge to the grandmaster.

D. Announce message Format

The original design of the IEEE 1588 standard, in which the `ANNOUNCE` message wraps together all of the node's specifications, is tailor made for distribution of cryptographic certificates. The required change to the message's structure is only the addition of two extension fields, as can be seen in [9]. Note that the second field is the management's signature over all of the `ANNOUNCE` fields.

Technical Note: During the BMC algorithm, after a slave validates a master candidate's certificate once, it needs only to compare its fields and signature with every newly received `ANNOUNCE` message from the same node. Only in case the certificate parameters change is there a need to actually perform a cryptographic verification check on it again. This means that the expectation of the overhead cost in signing and verifying these messages is practically zero: The management node needs to sign the certificate once, the master node only needs to copy it from its configuration, and the slave node needs only check it once per BMC-elected master.

E. Sync and Follow Up message Format

The `SYNC` message stayed unchanged in our scheme, due to the fact that we mandate the `FOLLOW_UP` message to allow hardware-based timestamping. The `FOLLOW_UP` message was only extended with the signature at its end, a signature that is calculated over all of the message.

F. Delay request and response

In our implementation we chose to keep the `DELAY_REQUEST` and `DELAY_RESPONSE` message untouched, i.e., not to sign the `DELAY_RESPONSE` message.

The main argument against the signing of these messages, is the rather low benefit one can gain from it. The `sequenceId` changes practically closed all attacks from Out-Of-Band (OOB) adversaries, while even the gap remaining (inband attacks) can easily be handled by deploying several sanity thresholds on the received timestamps, as suggested in [16], [1] and [12].

G. Performance

Our experiments were done using the *DeterLab* project environment, on which we used computers with specs of: 4 X Intel(R) Xeon (R) CPU X3210 @ 2.13Ghz, 4096 Bytes cache. According to our measurements, the average cost of signing the `FOLLOW_UP` message was only 0.41 msec, and the cost of verifying that message was only 0.17 msec. This means that even at a high rate of 128 messages per second, suitable for the telecom profile, the cryptographic work done by the CPU is negligible.

These positive results show that the lightweight bandwidth of the PTP protocol makes it unnecessary to use special hardware for the cryptographic layer, as was suggested in [17] and implemented in [7].

H. Comparison to Annex K and State of the Art

There is almost no connection between our security proposal to the original Annex K. The appendix uses a symmetric cryptographic solution to establish unicast trust relationships between two PTP nodes. This means that the appendix can not handle an *Insider* threat, and is specifically vulnerable to the attacks described in III-A5 and III-B1.

Our solution overcomes these problems due to its asymmetric nature: we suggest only a one way trust relation: slaves authenticate the master. Together with authenticated BMC algorithm, this enables a lightweight multicast solution, that is specifically tailored to the PTP protocol.

Recalling the 4-Pronged approach of [21] we see that our security extension has all of the needed qualities, and they are achieved in a unique way:

- Message integrity - asymmetric signature (EdDSA)
- Anti-Replay - enlarged `sequenceId` field
- Security association - binding the network identity to the clock ID, and using the clock ID as the security ID
- Key Distribution - predefined management public key, and predefined master-candidates certificates

We achieved additional goals, yet to be addressed:

- Node Authorization - signed `ANNOUNCE` certificates
- Challenge Response - using the `sequenceId` field, as a challenge for the delay/resp messages

And last, we suggest to deploy a standard and secure management protocol (e.g. SNMP), as a replacement to the PTP specific management.

I. Remaining Gaps

After ensuring the correctness of the BMC algorithm, the authenticity of the management functionality and the validity of the protocol's timestamps, there are still some remaining gaps. These remaining gaps are against the 2 all powerful insider adversaries, and against the In-Band adversary. A message delay attack will result in the same effect as encoding the same delay into the message's timestamp field. Suggested mitigations against such attack involve the deployment of multiple network paths, as in [13], [23] and [4], or using a threshold defense mechanism.

V. Conclusions and Future Work

In our work we have shown a detailed threat analysis of the Precise Time Protocol, including several new attacks and mitigations. We demonstrated the main vulnerabilities of the protocol: naive BMC leader election, unverified master timestamps, and lack of management authentication. Our analysis also shows that the protocol's security Annex K falls short of handling various threats resulting from an *Insider* adversary.

We suggest to deprecate the rather complicated security extension, in favor of our proposed PTP-specific defense scheme. Instead of moving towards a more complex symmetric-key cryptographic solution, we propose a public-key-based security solution specially tailored to the PTP protocol. Our solution is based on the basic difference between the 2 protocol roles: grandmaster and ordinary slave, a difference yet to be addressed by prior state of the art proposals.

We implemented all our attacks and cryptographic solutions and found that the modified protocol has excellent, practical performance on standard of-the-shelf computers. We believe that these additional security countermeasures will help to provide a complete security extension to the protocol, thus making the IEEE 1588 standard a more secure and robust network solution.

Acknowledgement

We thank D.J. Bernstein and Tanja Lange for suggesting the use of EdDSA and the Ed25519 curve in our implementation.

References

[1] M. Anyaegbu, Cheng-Xiang Wang, and W. Berrie. A sample-mode packet delay variation filter for IEEE 1588 synchronization. In *IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS)*, pages 1–6, Nov 2012.

[2] Terry Benzel. The science of cyber security experimentation: the DETER project. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 137–148. ACM, 2011.

[3] Daniel J Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *Journal of Cryptographic Engineering*, 2(2):77–89, 2012.

[4] M. Dalmas, H. Rachadel, G. Silvano, and C. Dutra. Improving PTP robustness to the Byzantine failure. In *IEEE Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, pages 111–114, Oct 2015.

[5] B. Dickerson. Observations on secure PTP, 2014. P1588 Working Group, https://ieee-sa.centraldesktop.com/1588/file/30928204/.

[6] L. Ellegaard. PTP security using MACsec, 2014. P1588 Working Group, https://ieee-sa.centraldesktop.com/1588/file/33390811/.

[7] H. Flatt, J. Jasperneite, and F. Schewe. An FPGA based cut-through switch optimized for one-step PTP and real-time ethernet. In *IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS)*, pages 7–12, Sept 2013.

[8] S. Fries. Proposal for IEEE 1588v3 security: Definition of a security TLV, 2014. P1588 Working Group, https://ieee-sa.centraldesktop.com/1588/file/31645125/.

[9] Eyal Itkin and Avishai Wool. A security analysis and revised security extension for the precision time protocol, 2016, arXiv:1603.00707 [cs.cr].

[10] K Lee, John C Eidson, Hans Weibel, and Dirk Mohl. IEEE 1588-standard for a precision clock synchronization protocol for networked measurement and control systems. In *IEEE*, volume 1588, 2008.

[11] J. Mirkovic, T. V. Benzel, T. Faber, R. Braden, J. T. Wroclawski, and S. Schwab. The DETER project: Advancing the science of cyber security experimentation and test. In *IEEE International Conference on Technologies for Homeland Security (HST)*, pages 1–7, Nov 2010.

[12] T. Mizrahi. A game theoretic analysis of delay attacks against time synchronization protocols. In *IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS)*, pages 1–6, Sept 2012.

[13] T. Mizrahi. Slave diversity: Using multiple paths to improve the accuracy of clock synchronization protocols. In *IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS)*, pages 1–6, Sept 2012.

[14] Tal Mizrahi. Time synchronization security using IPsec and MACsec. In *IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS)*, pages 38–43. IEEE, 2011.

[15] N. Moreira, J. Lazaro, J. Jimenez, M. Idirin, and A. Astarloa. Security mechanisms to protect IEEE 1588 synchronization: State of the art and trends. In *IEEE Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, pages 115–120, Oct 2015.

[16] T. Murakami, Y. Horiuchi, and K. Nishimura. A packet filtering mechanism with a packet delay distribution estimation function for IEEE 1588 time synchronization in a congested network. In *IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS)*, pages 114–119, Sept 2011.

[17] Cagri Onal and Hubert Kirrmann. Security improvements for IEEE 1588 Annex K: Implementation and comparison of authentication codes. In *IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS)*, pages 1–6. IEEE, 2012.

[18] W. Owczarek. Deploying PTP as an enterprise service: Issues, challenges and design considerations. In *IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS)*, pages 77–82, Sept 2013.

[19] Adrian Perrig, Dawn Song, Ran Canetti, JD Tygar, and Bob Briscoe. Timed efficient stream loss-tolerant authentication (TESLA): Multicast source authentication transform introduction. RFC 4082, 2005.

[20] Lukas Prokop. EdDSA notes. http://lukas-prokop.at/proj/eddsa/.

[21] Subcommittee S: Security SC standing document v5, 2015. P1588 Working Group, https://ieee-sa.centraldesktop.com/1588/file/39406935/.

[22] PTPd github. https://github.com/ptpd/ptpd.

[23] A. Shpiner, Y. Revah, and T. Mizrahi. Multi-path time protocols. In *IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS)*, pages 1–6, Sept 2013.

[24] D. Sibold. Status of network time security (NTS): Applicability for PTP, 2015. P1588 Working Group, https://ieee-sa.centraldesktop.com/1588/file/37027677/.

[25] W. Stallings. SNMPv3: A security enhancement for SNMP. *IEEE Communications Surveys*, 1(1):2–17, 1998.

[26] Jeanette Tsang and Konstantin Beznosov. A security analysis of the precise time protocol (short paper). In *Information and Communications Security*, pages 50–59. Springer, 2006.

[27] Hans Weibel. Technology update on IEEE 1588: The second edition of the high precision clock synchronization protocol. *Embedded World*, 2009.

[28] WolfCrypt. https://www.wolfssl.com/wolfSSL/Products-wolfcrypt.html.