Guards and Watchdogs in One-Way Synchronization with Delay-Related Authentication Mechanisms

Martin Langer, Kristof Teichel, Kai Heine, Dieter Sibold and Rainer Bermbach

Full Citation of the original article published by IEEE:

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, 2019, pp. 1-6.
doi: 10.1109/ISPCS.2019.8886633

Available at:
https://doi.org/10.1109/ISPCS.2019.8886633

Guards and Watchdogs in One-Way Synchronization with Delay-Related Authentication Mechanisms

Martin Langer*, Kristof Teichel†, Kai Heine*, Dieter Sibold† and Rainer Bermbach*
*Ostfalia University of Applied Sciences, Wolfenbüttel, Germany
†Physikalisch-Technische Bundesanstalt, Braunschweig, Germany
Email: {mart.langer, ka.heine, r.bermbach}@ostfalia.de, {kristof.teichel, dieter.sibold}@ptb.de

***Abstract*—In this paper, we consider ways of using secondary "Watchdog" mechanisms to protect primary time synchronization protocols from single-source or single-channel errors. This approach is particularly interesting when the Watchdog mechanism has stronger cryptographic protection than the primary synchronization mechanism. We specifically discuss the case where the primary mechanism employs one-way communication and is secured with an authentication scheme based on delayed disclosure of cryptographic information. Further, we present results from experiments with an implementation combining such a primary mechanism with a secured two-way control mechanism, which lead us to overall recommend the approach.**

I. Introduction

Time synchronization mechanisms working purely with one-way communication from a time source to its recipients have proven susceptible to attacks that employ delaying of messages [1], even if those messages are cryptographically secured [2]. Furthermore, the Timed Efficient Stream Loss-tolerant Authentication (TESLA) protocol [3] and variants of it [4] offer convenient light-weight security for one-way synchronization protocols, but can be completely bypassed by proper use of such delay attacks [5], [6]. In this paper, we thus pursue the task of finding convenient ways to mitigate such attacks and derive additional security guarantees for synchronization mechanisms based on one-way communication.

Several network-based synchronization protocols [7]–[9] and at least one synchronization mechanism [10] based on a Global Navigation Satellite System (GNSS) either already employ some variant of TESLA, or their current specification is proposing it. One standard draft [11] mentioned its use in a past version but has since opted to not cover one-way communication at all. Overall, the topic appears highly relevant for existing and emerging clock synchronization specifications.

The way that the potential attacks work is not only quite intricate, but also very specific to exactly the scenario where one-way communication is secured with a TESLA-like mechanism (i. e. a security mechanism that employs delayed disclosure of some cryptographic data to achieve asymmetric authenticity guarantees). We believe this to be a reason why the problem is sometimes ignored: it is not so much a weakness of TESLA itself as an issue of this specific application of it, and therefore only it occurs to very few users of it.

Partially, the project received funding from the EMPIR programme co-financed by the Participating States and from the European Union's Horizon 2020 research and innovation programme.

Occasionally, the problem is assumed solved or irrelevant due to properties of the communication environment, e. g. because it is hard for an attacker to acquire the necessary position as a man-in-the-middle (MITM) in a closed network such as is typically used for the Precision Time Protocol (PTP) [9], and even harder in a GNSS communication network. We believe that such an attitude is problematic, since it relies on an assumption about the commitment of the attacker. The effort required can also change over time, as it recently has been shown e.g. for jamming and spoofing of GNSS signals. Some solution attempts exist [2], [6] that are limited to protocols with built-in ways to add secured two-way synchronization into the message flow. While the solution may be efficient in those cases, there exist other scenarios in which the requirement of built-in two-way communication cannot be assumed (such as GNSS networks), or where the requirement to have everything available in-protocol causes an undesirable amount of additional complexity (such as in the case of PTP).

Our first contribution consists of a general approach to the problem described above, namely to combine different synchronization mechanisms, one with higher precision as the primary synchronization mechanism and another with stronger security guarantees as a control mechanism for the first. We supplement this with a list of concrete suggestions for candidate protocols for both roles. Our other main contribution consists of a presentation of results from a first implementation of our suggestion where the primary synchronization protocol is represented by NTP in broadcast mode. The secondary mechanism is represented by a two-way Network Time Protocol (NTP) [12] association secured with the Network Time Security (NTS) specification [11]. Hereinafter, NTS always means *NTS-secured NTP*.

The remainder of this paper is structured as follows: Related work is covered in Section II. Section III provides an introduction to the differences between one-way and two-way synchronization and to the TESLA protocol. Section IV introduces vocabulary for and discusses the general approach to increasing security via multi-protocol synchronization. Section V presents results on the practical aspects of using delayed-disclosure based security mechanism in synchronization and on the benefits of worst-case estimations for security in such scenarios. Section VI concludes the paper.

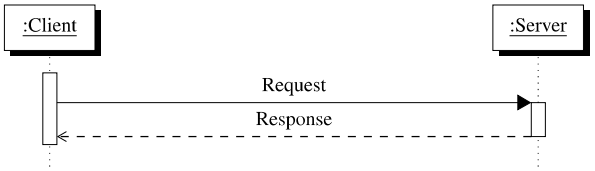

Fig. 1. Schematic of a two-way exchange, where the age of a request provides the client with a reliable upper bound for the age of the matching response.

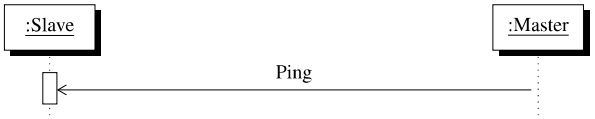

Fig. 2. Schematic of a one-way exchange, where the potential age of a message is not bounded by any information obtainable by its recipient.

II. Related Work

We first saw the idea of using a Watchdog protocol (and naming it such) in [13], where NTP is used as a control mechanism for a PTP infrastructure. The context in that work relates to redundancy and credibility checking rather than security, but prompted us to expand on it. Similar methods in PTP for mitigating attacks by so-called *guard clocks* can also be found in [14]. We also drew some inspiration from [15], especially from the way that it breaks with the notion of one device managing a single clock via a single synchronization mechanism. The component described as the "Quality of Time Core" in [15] directly inspired the idea of what we call a Coordinator. In both cases, we expand the ideas to cover cases where cryptographic security is involved.

There is an amount of work regarding the use of TESLA-like mechanisms to secure one-way time synchronization. Some of it can be found in specification documents [7]–[10], relating to this work by potentially being affected by it. Some of it exists in the form of security analyses [2], [5], [6], and can be seen as a precursor for this work (none of it considers out-of-protocol additions to the primary synchronization mechanism). Analyses in [16] also showed fundamental weaknesses in one-way time synchronization, which were considered in this paper.

III. Preliminaries

This section provides a discussion regarding the most impactful differences between two-way and one-way synchronization for our context, as well as an introduction to how the TESLA protocol operates.

A. One-Way versus Two-Way Time Synchronization

Generally speaking, network-based time synchronization is most often achieved via either one-way or two-way communication. Two-way communication mostly implies a client-server model where client requests are answered by server responses, as depicted in Fig. 1. Using the fact that a response is necessarily newer than its matching request, the client can calculate a value for the clock offset with a maximum error of half the network round-trip between request and response [17]. On the other hand, one-way communication usually implies a master-slave model, where the master periodically sends out messages to its slaves, as depicted in Fig. 2. Since there is no guarantee as to the age of a message, one-way scenarios offer no reliable upper bound for the error of the slave's calculation.

The presence and absence, respectively, of a guaranteed upper bound for the calculation error in two-way and one-way communication scenarios does not change when traffic is cryptographically secured. On the one hand, this means that if a two-way time synchronization exchange is properly secured, authentication strictly yields absolute certainty about the time offset from the server within the derived error bound. In scenarios where only one-way communication is used, on the other hand, there is no such guarantee: even if a message is certain to have originated from a given master, the calculated offset can still have an arbitrarily large error (since the message could have been delayed arbitrarily long).

Another crucial implication of the difference between one-way and two-way communication for security measures relates to scenarios where connection state on the server/master side is to be avoided (which is usually the case for time servers). In two-way communication scenarios, the connection state can be stored on the client and be made available to the master in every request (see [11], [18], [19] for examples of this approach). In pure one-way communication scenarios, however, such constructs are not feasible and thus asymmetric key establishment needs to be employed. Since classic asymmetric cryptography with algorithms allowing for public and private key pairs is too computationally expensive in most synchronization contexts, some protocols turn to authentication mechanisms based on delayed disclosure of cryptographic material, such as TESLA (see [7], [8], [10]).

B. An Introduction to TESLA

The TESLA protocol [3] is designed to secure one-way communication. For this purpose, the server sends the data to be protected in defined intervals at defined points in time. The length ΔT of those intervals is determined by the server and remains constant for the duration of the communication. A Message Authentication Code (MAC) over the payload forms the protection. However, the server always sends the associated symmetric key in a later interval (see Fig. 3), delayed by a disclosure delay d.

Due to the deferred transmission, the attacker is incapable of faking packets or injecting new ones. The keys used by the server at each interval are part of a key chain that the server generates upfront before the broadcast data transmission starts. The length of that chain determines the maximum duration of the current broadcast session. The cryptographic protection in TESLA is based on the irreversibility of the key generation. With any key from the chain, all previously used keys can be calculated until the end of the list. This provides the client with a great robustness against packet loss, because it can recover keys. However, the following keys as well as the source key of the chain cannot be determined. Clients start a bootstrap to obtain the necessary connection parameters like ΔT and d.

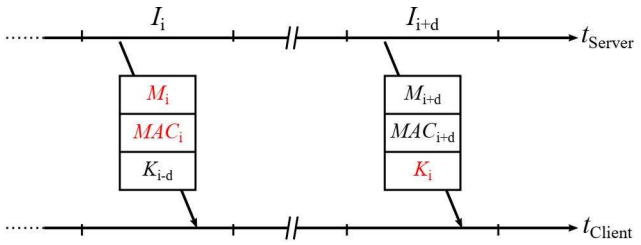

Fig. 3. Concept of the TESLA secured one-way communication

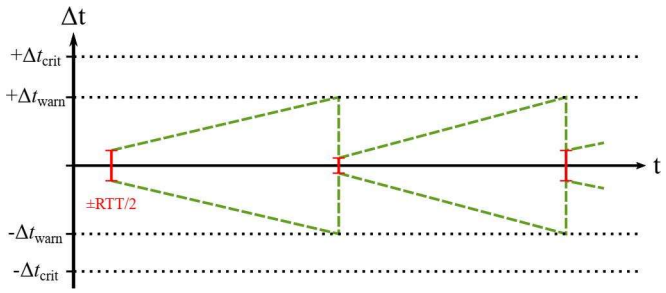

Fig. 4. Trusted Time Band (TTB)

In this course, the authentication of the server takes place, as well as the determination of the typical round-trip time (RTT). Since the bootstrap is a secure two-way communication, it is also possible to perform the initial synchronization of the clock.

IV. Increasing Security with Combinations of Synchronization Mechanisms

In this section, we elaborate on alternatives to built-in options for two-way communication, as well as advantages and disadvantages of employing them.

A. Terminology for Multi-Protocol Synchronization

In order to set up the discussion about multi-protocol techniques and security, we introduce a number of concepts and our nomenclature for referring to them.

1) Watchdogs: We refer to any secondary synchronization protocol as a *Watchdog* if it is somehow used as a control mechanism for a primary protocol (as opposed to as a mechanism to directly control and discipline a clock). A Watchdog can in particular ensure that existing security thresholds for time offsets posed by the primary protocol (such as in the case of TESLA-like mechanisms) are not exceeded, in which case the primary protocol gains a more reliable authenticity guarantee and inherits the maximum offset guarantee of the Watchdog.

2) Coordinators: In order for a Watchdog scheme to be efficient, it is essential to know how often and at which times during the runtime of the primary protocol the Watchdog needs to be put on duty. Employing it too slowly or not often enough can result in failure to obtain the desired guarantee, whereas excessive utilization can subvert the desired low effort of running a one-way mechanism in the first place. To make intelligent decisions about these issues, there needs to be an entity (program, operating system, external machine running either of the two, or even a person) with the capability to both control and evaluate results from the primary protocol as well as the Watchdog. We say that such a machine or program is channel-aware, and call it a *Coordinator*.

3) Trusted Time Bands and Guards: The final two concepts that we need to introduce for the discussions in the rest of this paper relate to how clock offset guarantees behave over time. Say that a reliable offset measurement was made at t_0, that it puts our local clock at offset δ and has an upper bound of ϵ for its error, and that we know an upper bound θ for our clock's drift rate. Then at time $t_1 > t_0$, we can deduce our offset to lie in the interval $[\delta-\epsilon-\theta\cdot(t_1-t_0), \delta+\epsilon+\theta\cdot(t_1-t_0)]$. If we regard t_1 as a function of time, we obtain a trapezoid shape spanning from our last reliable measurement into the future. We refer to this shape as a *Trusted Time Band (TTB)*, which is illustrated in Fig. 4 (green line). Assume that our TTB is based on measurements from a Watchdog and that we have determined a critical offset threshold Δt_{crit} posed by our primary protocol. Then we might be able to derive a warning threshold Δt_{warn} at which triggering the Watchdog will prevent that Δt_{crit} is ever exceeded. We refer to any algorithm that derives such a warning threshold from a critical threshold as a *Guard*.

B. Multi-Protocol Synchronization and TESLA Security

Based on results from [2] and considerations of initial synchronization data (and potentially runtime data as well), we can relate the concepts introduced above to the challenges regarding TESLA-secured one-way synchronization. In particular, it is possible for a Coordinator to employ a TESLA-secured one-way synchronization protocol and to make reliable worst-case estimations about the maximum offset that could have occurred at any given point during runtime. With this, the Coordinator can apply a Guard and trigger a Watchdog to guarantee that offsets critical for TESLA security are never exceeded. This seems particularly relevant to GNSS synchronization, since at least Galileo has an operational mode (the OSNMA [10]) which uses TESLA to secure one-way communication and can be used for time synchronization, but cannot readily be extended with two-way communication.

1) Protocol Candidate Suggestions: At this point, we would like to name a few candidates for both the primary protocol and the Watchdog role, where we feel that realizations could be especially useful. As primary protocols, we recommend any mechanism (secured or unsecured) based on GNSS, PTP or White Rabbit synchronization. All of them have high relative potential for obtainable accuracy and precision while often offering no security, or security based on one-way communication with the drawbacks discussed above. For an almost universal candidate for a Watchdog protocol, we suggest NTP, specifically secured variants of its client-server mode. Not only is NTP ubiquitous in its usability, but there exist

several modern iterations of light-weight cryptographic security mechanisms [11], [18], [19] for its two-way (client-server) mode of operation. When one of these approaches is used, the client obtains guaranteed authenticity for the response message, and thereby a guarantee about its measured offset, whose error cannot exceed half of the network roundtrip time. This upper bound is usually in the range of milliseconds, which is sufficient for most of what a TESLA-protected primary protocol might require from its Watchdog.

2) Concrete Implementation Suggestions: After considering the concepts above and their implications, we derive a number of concrete suggestions for setups of pairs of protocols, where one is used as a primary synchronization protocol and the other is used as a Watchdog.

- We suggest to use the NTS-secured NTP protocol [11], [12] as a Watchdog for a primary generic internet-based TESLA-secured broadcast synchronization protocol. We have implemented and tested something like it, where we we use broadcast NTP with added TESLA protection in extension fields as the primary protocol (see Section V).
- Separately, we suggest to use GNSS-based TESLA-secured synchronization (e. g. Galileo's OSNMA service) as a primary mechanism with an NTS Watchdog to ensure the offset stays within security requirements.
- Following up on the previous suggestion, we further suggest to re-interpret its resulting overall protocol as a Watchdog itself, and to use it to add security to a primary synchronization mechanism that is made up of (most likely unsecured) high-precision GNSS or White Rabbit synchronization.

For all three suggestions, we generally advise to start with runtime-independent worst-case analysis (e. g. following [2], [6]), but to experiment, with additional analysis at runtime, using runtime-generated data to allow for more generous worst-case calculations in the sense that they allow sparser use of the Watchdog protocol.

V. Implementation and Test Results

This chapter describes our realization of the combination of a TESLA-secured primary protocol and a Watchdog, both controlled by a Coordinator.

A. Delayed vs. Immediate Authentication

In TESLA, the protection of messages can be performed in two different ways. In the traditional variant (*Delayed Authentication*, see Fig. 3), the server transmits a message together with the associated MAC to the clients. No immediate verification of the message is possible, because the server transmits the necessary key in a later time interval after the disclosure delay d. If TESLA is used to synchronize the time, a dead time behavior occurs, since received time information can only be used after the delay d. Depending on the control system, this can lead to an overshoot of the time offset [5]. In this TESLA variant, clients must store the message and the MAC until the corresponding key is received. The strong robustness against packet loss, due to the recoverability of already published keys is an important advantage. However, the potential vulnerability of the client by packet flooding and the associated processor and memory usage presents a serious disadvantage.

In a second variant (*Immediate Authentication*), the server generates time messages and the corresponding MAC in advance, which are to be sent after the disclosure delay. Messages sent by the server in the current interval are transmitted together with the current interval key and the MAC of the future message. A client no longer has to store the whole message, but only the MAC. If the client receives the message at a later time, it immediately calculates the MAC and compares it with the previously received one. If both are identical, then the package is verified.

A disadvantage of the method is a reduced robustness of TESLA, because in case of packet loss the associated future message is no longer verifiable. A combined procedure as described in [20] could provide a remedy. As the server has generated time packets for future time intervals, they have to be sent exactly at the respective points in time. This produces an additional jitter, but it is significantly lower than the network jitter.

B. Difficulties and Solution

The problem of time synchronization by TESLA or TESLA-like mechanisms is a possible desynchronization of the client by an attacker and the subsequent break of the security features [2], [6]. In order to do this, an attacker delays packets in several phases. Clients react by setting back their own clock, which also shifts the time intervals defined by TESLA backwards. If the time difference between server and client exceeds the limit $t_{\text{crit}} = (d-1)\Delta T$, an attacker can transmit bogus packets to the client. This is possible because the attacker is in possession of the disclosed key that the client expects later because of the time offset. Thus, an attack is also possible with cryptographically secured packets.

The detection of the actual synchronicity is not possible due to the one-way communication, whereby both methods (delayed and immediate authentication) are affected. Solutions based on multiple time sources do not provide adequate protection, as they are also vulnerable in case of one-way communication [2]. In order to detect dangerous desynchronization, a periodic secure two-way synchronization is necessary. This contradicts the use of a broadcast connection for time synchronization.

The development of a Guard and a Coordinator as described in section IV-A3, is a solution. While the Guard registers changes to the clock, the Coordinator executes the Watchdog protocol when the Guard reports a critical level. This threshold depends on both parameters, the interval length ΔT and the disclosure delay d. To prevent that time t_{crit} is exceeded, a warning threshold can be defined by additionally using a pre-factor x_{warn}, before the critical threshold is reached: $t_{\text{warn}} = x_{\text{warn}}(d-1)\Delta T$ with $0 < x_{\text{warn}} < 1$. If an extensive time deviation is detected, the two-way synchronization can be used to correct the time and prevent a successful attack against TESLA.

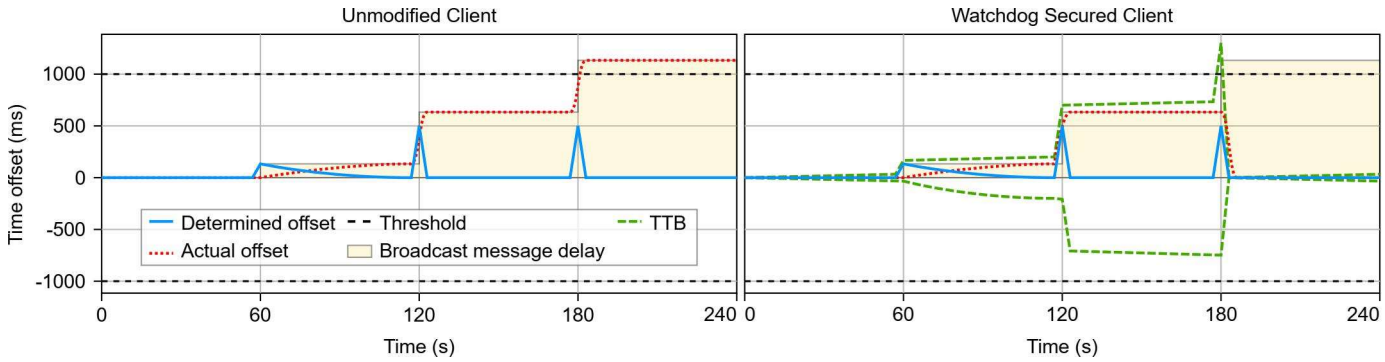

Fig. 5. Course of the Trusted Time Band (TTB) and trigger of the Watchdog protocol

C. Synchronization Guard and Worst-case Estimations for Guaranteed TESLA Security

Similar to the Watchdog protocol, the Guard does not require a built-in solution. In a simple implementation, the Guard runs as a separate function that observes the clock of the client and controls a *Trusted Time Band*. The TTB is a measure of uncertainty and represents a time range in which the current time of the client is guaranteed. Thus, a larger TTB means a greater uncertainty. As various factors affect the clock, the uncertainty in both positive and negative direction increases continuously. A Guard implementation provides the best protection with a *worst-case* configuration. Here, all external factors can only increase the TTB, even if logical corrections may reduce the uncertainty.

The first factor that affects the clock is a protected two-way time synchronization (e. g. via NTS [11]). This provides a secured time information and also a defined uncertainty, which is at $\pm RTT/2$ and thus specifies the starting range of the TTB. The second factor is the crystal in the hardware. Depending on the type and quality of the built-in oscillator, its wander causes a certain jitter to the given accuracy. Non-stabilized crystals also show large variations in temperature changes, which can cause a total deviation of over 250 ppm. The *frequency offset* that counteracts and compensates the wander of the oscillator, represents the third factor. This parameter can be changed by API functions of the operating system that allow the clock adjustment. The last factor is the hard setting of time in any direction (so-called *step time*). Finally, the channel-aware Coordinator executes the Watchdog protocol as soon as the uncertainty reaches a threshold defined by a Guard.

D. Implementation of TESLA-Secured One-Way Primary Protocol with Two-Way NTS Watchdog

To demonstrate the additional protection of a one-way protocol, a C++ implementation of TESLA with immediate authentication and an appropriate Guard were developed. The hardware setup used for the measurements includes two Ethernet-connected Raspberry Pi devices as broadcast server and client. The server continuously synchronizes to an external time server and distributes time data via TESLA and NTS. In addition to the TESLA protocol, the client is also equipped with the described Guard and acts as a Coordinator using NTS as a Watchdog protocol. It is also connected to the same external time server to determine the current offset to the broadcast server. In order to better illustrate the operation of Coordinator, Guard and Watchdog, appropriate test parameters were chosen.

Results of this example measurement are shown in Fig. 5. The diagrams show the effects of an attack on a client without a watchdog mechanism (left) and with a watchdog mechanism (right). At the beginning of the measurement, the TTB based on two-way synchronization is small (green line). After 60 seconds, the attacker applies a small delay that the client corrects continuously via the clock-adjust mechanism. The determined time offset (blue line) moves towards zero, while the uncertainty and thus the TTB (green line) grows. At 120 seconds, the attacker increases the delay, causing the client to perform a *step time* to correct the clock. While the client assumes itself synchronous, the actual time offset (red line) shows a strong deviation. After a total time of 180 seconds, the attacker again increases the delay and causes an unsecured client to desynchronize beyond the threshold. If the limit value is x_{crit}, the break of TESLA is possible and the client would accept any time of the attacker. With a Watchdog-secured client, however, the attack is detected. All clock changes have strongly increased the TTB, which finally reaches the upper bound. The Coordinator registers when the threshold defined by the Guard is exceeded and starts the Watchdog protocol. Due to the secured two-way communication and the guaranteed accuracy of $\pm RTT/2$, the client will be synchronized correctly and the TTB is adjusted to a low value once again.

The security of the TESLA communication can be significantly increased by using a Coordinator, Guards and a Watchdog protocol. For example: When using the parameters $\Delta T = 6$ and $d = 2$ as described in [6, p. 7-8], the critical desynchronization limit of the client is $t_{\text{crit}} = 6\,s$. Therefore, an unprotected attack on TESLA would be successful after approximately 48 seconds if the client adapts the clock for larger offsets via *step time* without further condition. The only possible protection is a secured and periodic two-way time synchronization after every 48 seconds at the latest.

However, Coordinator and Guard as described above detect

and prevent this attack. Furthermore, the execution of the Watchdog protocol can be heavily postponed without risk. The time at which the Watchdog protocol is executed depends on Guard and TESLA parameters. With a pessimistically estimated oscillator wander of ±300 ppm, the execution of the Watchdog can be postponed up to 5.5 hours. Since the increase of the disclosure delay in case of *immediate authentication* does not affect the accuracy, the trigger time of the Watchdog protocol can also be significantly protracted. The change of the disclosure delay from $d = 2$ to $d = 5$ with the same interval length delays the execution of the Watchdog protocol up to 22.2 hours. In comparison to the 48 seconds execution period of the unprotected TESLA, proposed solution gains a huge benefit.

VI. Conclusion and Future Work

We investigated potential security benefits of combining multiple synchronization protocols. Specific attention was given to scenarios in which a one-way protocol with high precision and accuracy acts as the primary protocol and a two-way protocol with higher security acts as a Watchdog for the primary protocol. In particular, we focused on primary protocols that use TESLA-secured one-way communication in combination with a Guard and a Coordinator.

The idea of using combinations of protocols might seem cumbersome, and it does indeed cause an increase in complexity. However, until a completely new mechanism is introduced that achieves the precision and accuracy of well-tuned White Rabbit or GNSS synchronization while offering serious cryptographic security deriving two-way based offset error guarantees, there will still be benefits exclusive to combinations of protocols. Given the outcomes of our considerations and experiments, we overall recommend this approach.

There are a few items of future work that we can already name at this point. First, we will work on the development of a formal proof of the generic overall security properties of combinations of one-way primary protocols with two-way Watchdogs. Another step is to enhance the Guard to allow an improved adjustment of the *Trusted Time Boundaries*. This can further postpone the execution of the Watchdog protocol, without any danger of a successful delay attack.

It would further be interesting to extend or alter our implementation to accommodate for specific primary protocols that have no convenient built-in way to occasionally perform secured two-way synchronization, such as Galileo OSNMA, or PTP. We additionally encourage others to start working in this direction. Lastly, we would welcome considerations about how our concept of source-aware and channel-aware machines relates to existing concepts such as that of a "Timeline" [15]. It is our sincere hope that the entirety of the mentioned efforts will facilitate a significant increase in achievable security in high-precision time synchronization.

Acknowledgments

The authors would like to thank Stefan Milius for his input and ideas. Parts of this work have received funding from the EMPIR programme co-financed by the Participating States and from the European Union's Horizon 2020 research and innovation programme.

References

[1] T. Mizrahi, "Security requirements of time protocols in packet switched networks," IETF Secretariat, RFC 7384, 10 2014.

[2] K. Teichel, D. Sibold, and S. Milius, "An attack possibility on time synchronization protocols secured with tesla-like mechanisms," in *Proc. Information Systems Security (ICISS)*, ser. Lecture Notes Comput. Sci., I. Ray, M. S. Gaur, M. Conti, D. Sanghi, and V. Kamakoti, Eds., vol. 10063, 2016, pp. 3–22.

[3] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "The tesla broadcast authentication protocol," *RSA Cryptobytes*, vol. 5, 2005.

[4] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "Spins: Security protocols for sensor networks," *Wireless networks*, vol. 8, no. 5, pp. 521–534, 2002.

[5] K. Teichel, D. Sibold, and G. Hildermeier, "Delayed authentication and delayed measurement application in one-way synchronization," in *2018 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, 09 2018.

[6] 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.

[7] K. Sun, P. Ning, and C. Wang, "Tinysersync: secure and resilient time synchronization in wireless sensor networks," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 264–277.

[8] X. Yin, W. Qi, and F. Fu, "Asts: An agile secure time synchronization protocol for wireless sensor networks," in *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*. IEEE, 2007, pp. 2808–2811.

[9] "Standard for a precision clock synchronization protocol for networked measurement and control systems," IEEE, Standard 1588. [Online]. Available: https://standards.ieee.org/develop/project/1588.html

[10] I. Fernandez-Hernandez, V. Rijmen, G. Seco-Granados, J. Simon, I. Rodríguez, and J. David Calle, "A navigation message authentication proposal for the galileo open service," *Navigation - Journal of The Institute of Navigation*, vol. 63, 03 2016.

[11] D. Franke, D. Sibold, K. Teichel, M. Dansarie, and R. Sundblad, "Network Time Security specification for the Network Time Protocol," Internet Engineering Task Force, Internet-Draft draft-ietf-ntp-using-nts-for-ntp, 02 2019, work in Progress.

[12] D. Mills, J. Martin, J. Burbank, and W. Kasch, "Network time protocol version 4: protocol and algorithms specification," Internet Requests for Comments, IETF Secretariat, RFC 5905, 06 2010, https://tools.ietf.org/html/rfc5905.

[13] P. V. Estrela, S. Neusüß, and W. Owczarek, "Using a multi-source ntp watchdog to increase the robustness of ptpv2 in financial industry networks," in *2014 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, Sep. 2014, pp. 87–92.

[14] B. Moussa, M. Debbabi, and C. Assi, "A detection and mitigation model for ptp delay attack in an iec 61850 substation," *IEEE Transactions on Smart Grid*, Sep. 2018.

[15] F. Anwar, S. D'Souza, A. Symington, A. Dongare, R. Rajkumar, A. Rowe, and M. Srivastava, "Timeline: An operating system abstraction for time-aware applications," in *2016 IEEE Real-Time Systems Symposium (RTSS)*, Nov 2016, pp. 191–202.

[16] L. Narula and T. Humphreys, "Requirements for secure clock synchronization," *IEEE Journal of Selected Topics in Signal Processing*, 2017.

[17] J. Levine, "A review of time and frequency transfer methods," *Metrologia*, vol. 45, no. 6, pp. S162–S174, 2008.

[18] F. Mkacher, X. Bestel, and A. Duda, "Secure time synchronization protocol," in *2018 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, Sep. 2018, pp. 1–6.

[19] B. Dowling, D. Stebila, and G. Zaverucha, "Authenticated network time synchronization," Cryptology ePrint Archive, Report 2015/171, 2015, http://eprint.iacr.org/2015/171.

[20] A. Perrig, R. Canetti, D. Xiaodong Song, and J. D. Tygar, "Efficient and secure source authentication for multicast," Jan 2001.