MAC-layer Security for Time-Sensitive Switched Ethernet Networks

Venesa Watson[1] Prof. Dr. Christoph Ruland[2] Dr. Karl Waedt[3]

Abstract: Security remains a key discussion point for industrial networks within critical infrastructure and Industry 4.0 (I4.0)/Smart Manufacturing infrastructures. While availability remains the chief security requirement for highest safety, integrity protection has become somewhat equal to availability in industry. Common integrity protection mechanisms, however, are not practical for the time-sensitive networks (TSNs) characteristic of I4.0 and critical infrastructures, where the time-critical and mission-critical transmissions cannot be negatively affected by the security overhead. To sufficiently protect and support TSNs, it is necessary to design an integrity protection scheme that provides lightweight security particularly at the OSI MAC-layer where the TSN protocols are defined. The development and testing of lightweight cryptographic algorithms provide one mean by which to achieve such an integrity protection, however, additional steps are needed to design and prove a suitable scheme. TSN-MIC is proposed as a viable scheme for MAC-layer security for TSNs in critical infrastructure and I4.0/Smart Manufacturing.

Keywords: critical infrastructure; industry 4.0; integrity protection; time-sensitive networks; OSI MAC-layer; lightweight cryptography

1 Introduction

Time-Sensitive Networks (TSNs) are those specifications that deploy services such as time synchronization, traffic categorization and traffic shaping, to support time-critical and mission-critical transmissions. For future infrastructures (I4.0/Smart Manufacturing), the Time-Sensitive Networking (TSNg) IEEE 802.1 sub-standards are earmarked as the protocols to provide these listed services. Specifically, OPC UA/TSNg is proposed, where OPC UA is a communication protocol to support interoperability between the interconnected systems – OPC UA is defined in IEC 62451. Similar TSN specifications include ARINC 664 Part 7 - Avionics Full-Duplex Switched Ethernet (AFDX), the IEEE 1722-2016 defined Audio-Video Transport Protocol (AVTP), which like OPC UA/TSNg uses the IEEE 802.1 TSNg sub-standards, and SAE AS6802 Time-Triggered Ethernet (TTE) from TTTech. PROFINET, which is a popularly used industrial protocol, also supports time-sensitive services. These TSNs are compared in [WS18]. Even though these specifications implement

[1] Universität Siegen, Lehrstuhl für Digitale Kommunikationssysteme, Hölderlinstraße 3, 57076 Siegen, venesa.watson@uni-siegen.de
[2] Universität Siegen, Lehrstuhl für Digitale Kommunikationssysteme, Hölderlinstraße 3, 57076 Siegen, Christoph.Ruland@uni-siegen.de
[3] Framatome GmbH, Cybersecurity Concepts & Architecture, Paul-Gossen-Straße 100, 91052 Erlangen

doi:10.18420/inf2020_32

their time-sensitive services in different ways, they share one key similarity. That is, all their time-sensitive services are implemented at the OSI MAC-layer. As such, it is critical to implement dedicated security at the MAC-layer. Traditional security mechanisms implemented at higher OSI layers offer limited to no protection at the lower OSI layers – in that, the design of the OSI model means that protocols at one layer are unaware of issues at another layer, and security at higher layers will not benefit the MAC layer. Additionally, traditional security mechanisms are considered too resource intensive for time- and mission-critical transmissions, especially in safety-critical infrastructures, such as nuclear power plants. With the introduction of lightweight cryptography algorithms, however, the opportunity is presented to develop viable security schemes for TSNs.

Time-Sensitive Network – Message Integrity Code (TSN-MIC) is one such security scheme that offers lightweight integrity protection designed specifically for MAC layer TSN services. Where TSN-MIC further differs from other MAC-layer security schemes (for example, IEEE 802.1X Port-Based Network Access Control, IEEE 802.1AE MAC Security (MACsec) and IEEE 802.10 Standard for Interoperable LAN/MAN Security (SILS)) is in the additional mechanisms that are included for improved security while still observing the performance requirements of time-critical transmissions. TSN-MIC is designed with an online key management and key change-over mechanism, with feedback mechanisms to detect and restrict the propagation of error related to intention and unintentional actions. This scheme is described further in the subsequent sections as follows: Section 2 discusses the TSN-MIC parameters; Section 3 describes how the parameters come together in the overall TSN-MIC concept; Section 4 provides a demonstration of the TSN-MIC scheme; and Section 5 provides a summary.

2 Security Scheme Parameters

The key parameters of the Time-Sensitive Network – Message Integrity Code (TSN-MIC) scheme are the lightweight cryptography algorithms for the generation of the MIC (also Message Authentication Code) and the key management protocol for the key lifecycle processes (key generation, key deactivation, key update and key change-over). For these parameters, only ISO/IEC standards are considered to ensure that non-proprietary, peer-reviewed algorithms from a trusted source are used.

For the MIC generation, ISO/IEC 29192-6 Chaskey-12 [IS17] is used. The Chaskey algorithm that takes an arbitrary-length message (m) that it processes in 128-bit blocks using an n-bit (128-bit) key to create a MAC (τ) of 128 bits or less. The efficiency of the Chaskey algorithm is observed in [Mo15], where the results indicate that Chaskey-8 (Chaskey with 8 rounds) is between 7 to 15 times faster than AES-128-CMAC on the microcontrollers used, while Chaskey-12 (Chaskey with 12 rounds) is 15% slower than Chaskey-8 on the same microcontrollers. TSN-MIC can be implemented with any selected algorithm, however, Chaskey is highlighted here as a viable lightweight cryptography with provable efficiency and security. In [MM14], additional advantages of Chaskey are given:

- **Cross-Platform Versatility** – all microcontrollers do not support variable-length bit rotations and bit shifts, by selecting some rotation constants to be multiples of 8 (i.e. Chaskey-8, Chaskey-16), this limitation is overcome.
- **Dedicated Design** – Chaskey is designed for 32-bit microcontroller architectures.
- **Key Agility** – By generating subkeys into state through simple XOR operation, Chaskey is more efficient than if using a key schedule.
- **Nonces are optional** – an algorithm is susceptible to an attack if the nonce is reused, Chaskey does not require a nonce, therefore avoiding the security issue.
- **Patent-Free** – Chaskey has no known patents or patent applications.
- **Provably Secure** – Chaskey is designed based on an Even-Mansour block cipher based on the permutation operation. The minimum data complexity is $\boldsymbol{D = 2^{64}}$ for Chaskey. A data complexity $\boldsymbol{D}$ below $\boldsymbol{2^n/2}$ avoids chosen plaintext attacks (internal collisions). A time complexity $\boldsymbol{T}$ below $\boldsymbol{2^n/D}$ avoids attacks with a practical time complexity. By restricting the total number of blocks to be processed under one key to 2^{48} blocks, this makes the Chaskey implementation resistant to known plaintext attacks.
- **Resistance Against Timing Attacks** – Chaskey is inherently secure against timing attacks, as the message length determines the total number of cycles.
- **Tag Truncation** – the best attack on Chaskey with short tags is tag guessing, but the algorithm is otherwise robust under tag truncation. The recommended tag size is $|\tau| \geq \mathbf{64}$ for typical applications.

For the key management, the schemes in ISO/IEC 11770 are considered. The ISO/IEC 11770-2 mechanism 6 for key establishment is eventually selected, as it supports key authentication, replay detection, key confirmation, key compromise impersonation and entity authentication [IS16]. ISO/IEC 11770-2 mechanism 6 describes a key generation mechanism between an initiating system and a responding system, with parameters (keying material, identifier and random number) critical to generating matching key pairs and for assuring the integrity of the messages exchanged between these two systems. Analysis of this scheme suggests that the parameters could be vulnerable to manipulations that could disrupt the key generation process and go undetected. As such, the ISO/IEC 11770-2 mechanism 6 is expanded to design a more robust key lifecycle management protocol. Fig. 1 and Fig. 2 illustrate the changes between the original ISO/IEC 11770-2 mechanism 6 and the TSN-MIC mechanism based on the former. In the TSN-MIC mechanism, the major changes include:

- Use of the Random number parameters as message tags to assure message integrity (steps 2 and 3).

- Use of a challenge-response mechanism to ensure that the matching key pairs are generated (steps 4 to 7).
- Online key update and key change-over which monitor key usage and triggers the initiating system to first checks for (and, where necessary triggers the generation of) a successor session key (*next* session key to replace the *current* session key), and the swaps the session keys (steps 8 and 9).
- A threshold for monitoring successive mis-matched MIC in the MIC verification process (step 10).
- Key revocation for expired and/or unusable key pairs (step 11).
- Integrity check for messages transmitted between the initiating system and the responding system (step 12)

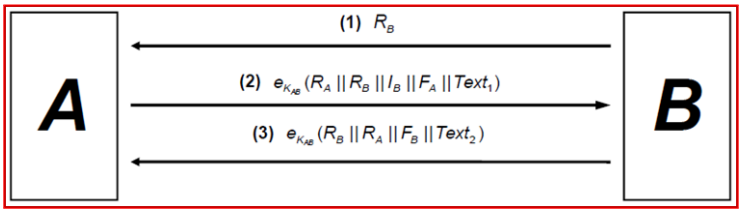

Fig. 1: ISO/IEC 11770-2 mechanism 6

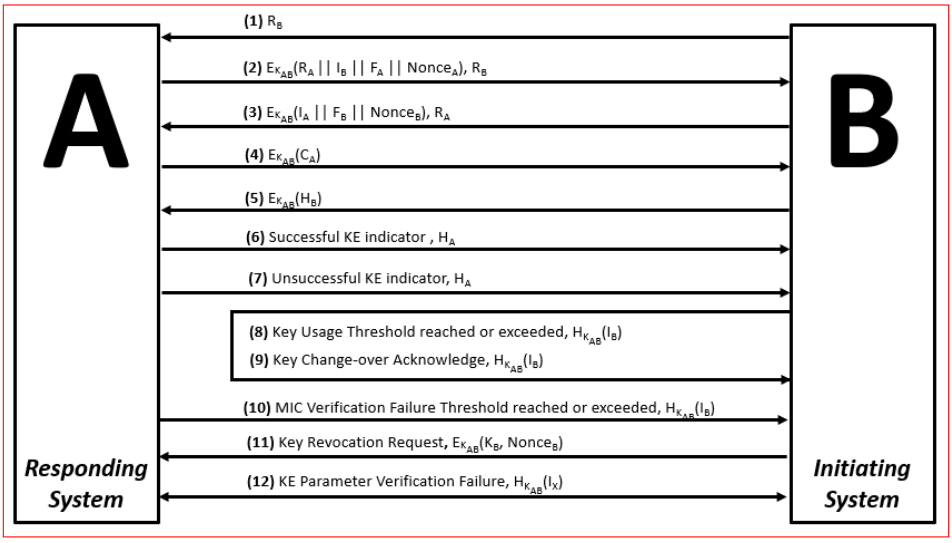

Fig. 2: TSN-MIC-based ISO/IEC 11770-2 mechanism 6

Both the use of Chaskey-12 and the TSN-MIC-based key management mechanism serve to ensure the efficiency and security of the overall TSN-MIC scheme. In the following section, the TSN-MIC is further described to illustrate how the given parameters are leveraged.

3 Security Scheme Concept

As noted, the TSN-MIC scheme is implemented at the OSI MAC-layer, just below the services/operations of the selected TSN protocol (e.g. AFDX, AVTP, TTE, etc.). Fig. 3 gives a depiction of this placement of the TSN-MIC operations (TSN-MIC_OP) below the TSN operations (TSN_OP). This design means that messages are first processed by the TSN-MIC operations before the messages are handled by the TSN operations- in that, for example, TSN-MIC operations for MIC verification is determined and non-compliant messages are dropped before the effort of the TSN operations are wasted. A more detailed description of the TSN-MIC operations is given below.

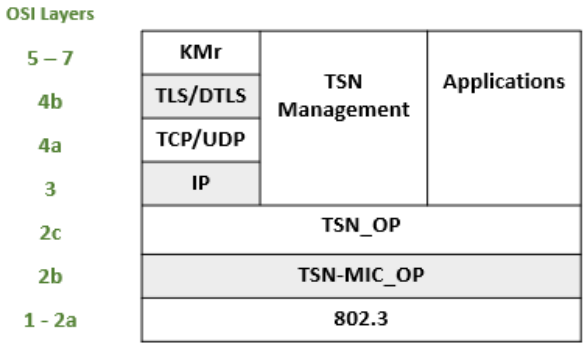

Fig. 3: TSN-MIC protocol stack

The TSN-MIC operations across a simple network (a Source End System, a TSN Switch and a Destination End System), there are seven (7) calculations. Namely, there are two main calculations that are repeated – that is three (3) Long Hash Calculations (LHCs) and four (4) Short Hash Calculations (SHCs). The LHCs are so-called as the hash (unkeyed) output of this calculation is always calculated over the payload of the frame, which is normally between 46 to 1500 bytes. The SHCs are so-called as the MIC (keyed) output is always calculated over the hash output of the LHCs, which is set at 16 bytes in this description of TSN-MIC but can be larger (recommended) or smaller. Across the *simple network*, these seven (7) calculations are as follows and illustrated in Fig. 4:

- **MIC Generation at Source End System**

  The ES generates frames that is processed by the TSN-MIC operation - one LHC followed by a SHC:

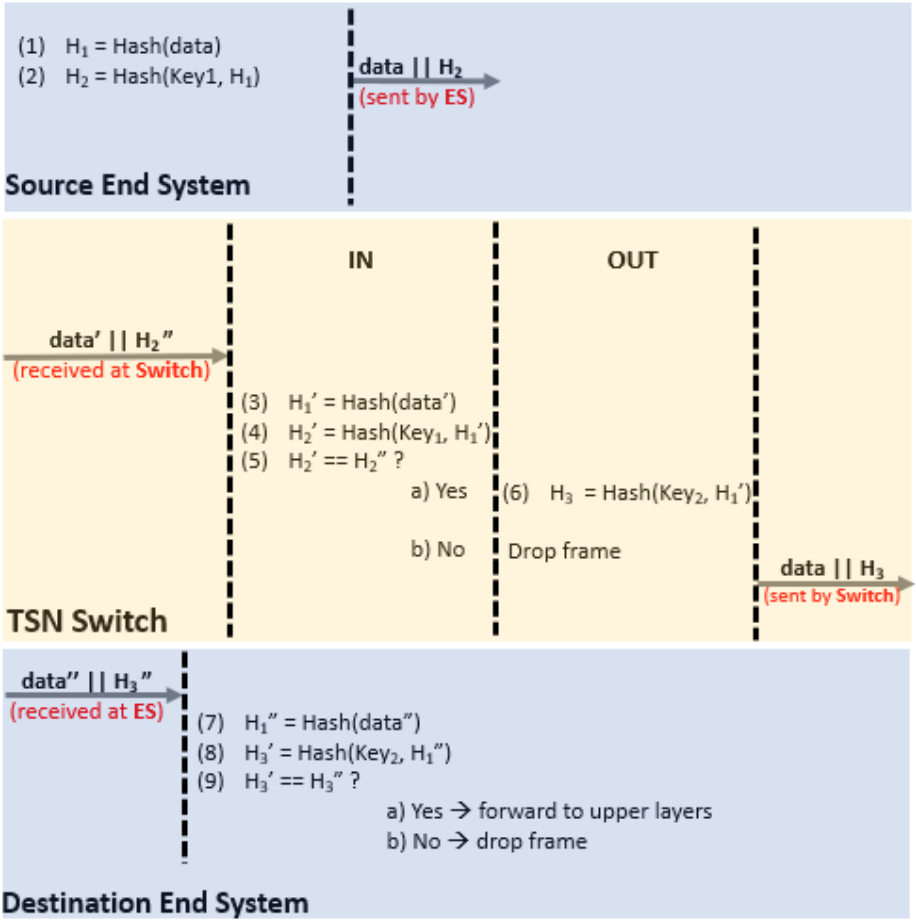

Fig. 4: TSN-MIC operations across a simple network

1. LHC: Generate an unkeyed hash over the data to output the hash H_1
2. SHC: Generate a keyed hash using the session key (Key_1) over the hash H_1 to output the MIC H_2

- **MIC Verification and Re-Generation at the Switch**

  At the TSN Switch, there is both a verification operation and a re-generation operation, involving one (1) LHC and two (2):

3. LHC: Generate an unkeyed hash over the data to output hash H_1'

4. SHC: Generate a keyed hash using the session key (Key1) over hash H_1' to output MIC H_2'
5. If MIC H_2' == MIC H_2", perform calculation (6), otherwise, drop the frame
6. SHC: Generate a keyed hash using the session key (Key_2) over hash H_1' to generate MIC H_3

- **MIC Verification at Destination End System**

  The TSN-MIC calculations at the Destination ES mirror that at the Source ES, but with the inclusion of a verification step:

7. LHC: Generate an unkeyed hash over the data to output hash H_1"
8. SHC: Generate a keyed hash using the session key (Key2) over hash H_1" to generate MIC H_3'
9. If MIC H_3' == MIC H_3", transmit data to upper protocol layers, otherwise, drop data

The initial session keys used to generate the MIC between the communicating systems are manually included, with subsequent keys being generated through the TSN-MIC key management mechanism. Each End System and TSN Switch is configured with two initial session keys (one for incoming messages and one for outgoing messages) that are used for calculating the MIC over the frame payload. Further, each End System and TSN Switch is configured with a master key that is used for the key establishment procedures to provide confidentiality for the keying material that is shared between the negotiating systems. The MIC calculations are done on a per link basis so that incoming messages and outgoing messages are respectively processed with the link-specific session keys. This means that the session keys and master keys can be distributed in a way to avoid the n^2 key distribution problem. For instance, a single session key or master key is not used for the entire end-to-end transmission of a message. At each traversing system, a different key is used. Therefore, each system does not need to have a copy of every key currently in use across the network, which eliminates the n^2 key distribution problem. Fig. 5 gives an illustration of how session keys and master keys are distributed in a TSN-MIC system.

With the parameters and concept determined, the next step is to implement and test the TSN-MIC scheme to assess its viability. This is discussed in the next section.

4 Security Scheme Implementation and Simulation

To test the concept, a software implementation was completed on a Banana Pi, while the efficiency of the TSN-MIC scheme was demonstrated using the OMNeT++ simulator. For the former, the Banana Pi was configured as a TSN Switch, modelled after the AFDX

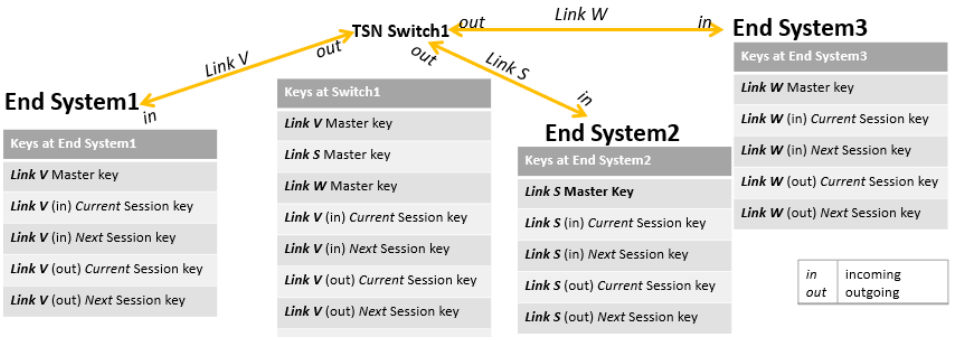

Fig. 5: Example TSN-MIC key distribution

specification. A laptop was then configured as an AFDX End System (ES), serving as both the source ES and the destination ES. Both devices were then configured to conduct the TSN-MIC operations as described earlier where the long hash calculations (LHCs) and short hash calculations (SHCs) are featured, as well as the key management protocol messages. This *simple network* is then activated to exchange the messages and to process them with the TSN-MIC operations (where appropriate, for example, key establishment messages are not processed by the TSN-MIC LHCs and SHCs). Outputs of this simple network in operation are given below. Fig. 6 gives a Wireshark® screenshot these messages being exchanged between the Banana Pi TSN Switch and the End Systems (laptop). Fig. 7 to Fig. 8 are example outputs of the key establishment process, while Fig. 9 shows messages that demonstrate the key update and key change-over process. In the former, the toggle bit is a reserved portion of the SHC MIC output that is flipped between “0” and “1” to indicate a session key update and change-over at the initiating system and to trigger the same at the responding system.

No.	Time	Source	Destination	Protocol	Length	Info
295	461.215612	192.168.178.92	192.168.178.90	UDP	502	2000 → 1045 Len=444
297	463.213079	192.168.178.92	192.168.178.90	UDP	502	2000 → 1045 Len=444
299	465.221103	192.168.178.92	192.168.178.90	UDP	502	2000 → 1045 Len=444
300	467.232230	192.168.178.92	192.168.178.90	UDP	502	2000 → 1045 Len=444
305	469.239657	192.168.178.92	192.168.178.90	UDP	502	2000 → 1045 Len=444
307	471.255473	192.168.178.92	192.168.178.90	UDP	502	2000 → 1045 Len=444

Fig. 6: TSN-MIC message transmission between TSN Switch and End Systems

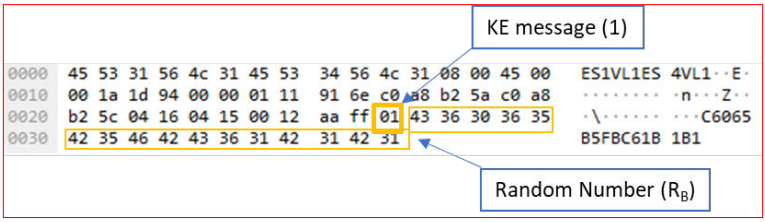

Fig. 7: TSN-MIC key establishment message (1)

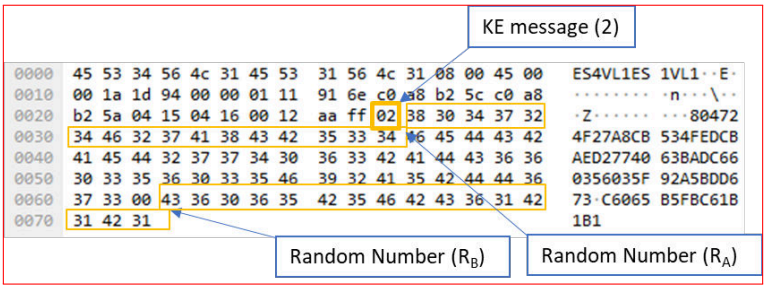

Fig. 8: TSN-MIC key establishment message (2) - unencrypted

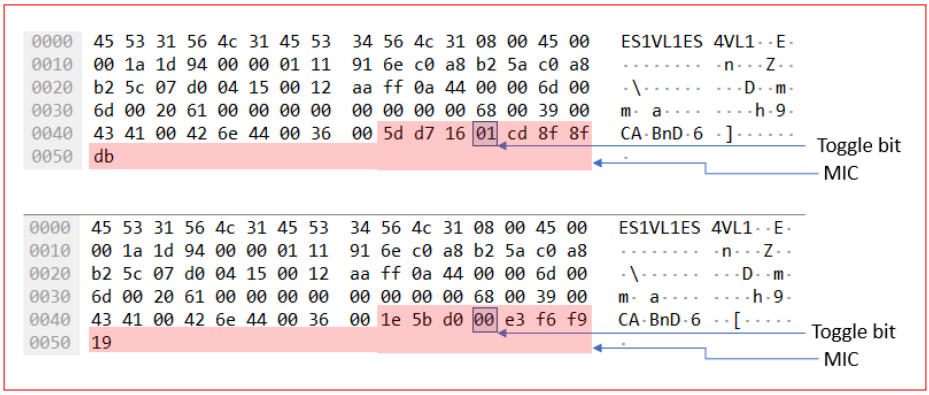

Fig. 9: TSN-MIC Key Update and Key Change-over with Toggle bit update

An OMNeT++ AFDX simulation was taken from [VH12] and modified to model the TSN-MIC design – in that, modules were added to represent the TSN-MIC LHCs and SHCs. Initially, theoretical values are derived to demonstrate the network efficiency and used as a benchmark to assess the efficiency of the TSN-MIC scheme in the OMNeT++ model. For this, the results for the performance of Chaskey-12 is taken from [MM14] and used as a reference point for the theoretical values. [MM14] demonstrates that the performance of Chaskey-12 is 10.5 cycles/byte (speed) and 84×10^6 cycles/second (processor speed) on the ARM Cortex-M3/M4; and at 25.4 cycles/byte (speed) and 48×10^6 cycles/second (processor speed) on the ARM Cortex-M0. In bytes per second, these performance values are respectively 8.0×10^6 on ARM Cortex-M3/M4 and 1.9×10^6 on ARM Cortex M0. In the TSN-MIC LHC, frames between 46 bytes and 1500 bytes are processed, while the TSN-MIC SHC will only ever process 16 bytes (maximum Chaskey MIC output). Based on [MM14], Chaskey-12 would then process 46 bytes in 5.98 μs to 24.38 μs, and between 195 μs to 795 μs for 1500 bytes. Tab. 1 gives the theoretical performance of TSN-MIC based on these values. Based on the theoretical calculations, on the ARM Cortex-M3/M4, the expected TSN-MIC delay is 8.06 μs to 197.08 μs for messages between 46 bytes to 1500 bytes, while on an ARM Cortext-M0, the TSN-MIC delay is 32.86 μs to 803.48 μs.

From this, it can be deduced that there is an overall expected increase of 1% to 35% in the transmission time when a message is processed by a combination of the TSN-MIC LHC and SHC as compared to a message of the same size being processed by a single Chaskey-12 calculation (Tab. 2).

Tab. 1: Theoretical performance of TSN-MIC calculations

Platform	Time per byte	TSN-MIC delay for 46-byte frame	TSN-MIC delay for 1500-byte frame
TSN-MIC Long Hash Calculations			
ARM Cortex-M3/M4	0.13 μs	5.98 μs	195 μs
ARM Cortex-M0	0.53 μs	24.38 μs	795 μs
TSN-MIC Short Hash Calculations			
ARM Cortex-M3/M4	0.13 μs	2.08 μs	2.08 μs
ARM Cortex-M0	0.53 μs	8.48 μs	8.48 μs

Tab. 2: Theoretical performance of TSN-MIC vs Single Chaskey-12 calculation

Platform	Chaskey-12 delay	TSN-MIC delay (1 LHC + 1 SHC)	Percentage Change
Ethernet frame of 46 Bytes			
ARM Cortex-M3/M4	5.98 μs	8.06 μs	+35%
ARM Cortex-M0	24.38 μs	24.38 μs	+35%
Ethernet frame of 1500 Bytes			
ARM Cortex-M3/M4	195 μs	197.08 μs	+1%
ARM Cortex-M0	795 μs	803.48 μs	+1%

The efficiency of the TSN-MIC security scheme is then assessed using the OMNeT++ AFDX model. First, the *QueryPerformanceFrequency* function is used to observe the average processing times TSN-MIC LHC and TSN-MIC SHC with the underlying Chaskey-12 algorithm. The average time taken for a TSN-MIC LHC is observed to be 26.6 ms of messages of 128 bytes, and 19.4 ms for a TSN-MIC SHC for messages of 16 bytes. Next, the OMNeT++ model was observed to determine the time taken for an end-to-end delivery, which is given as 10.88x10-6 simsec (1 simsec $\approx$ 4.867 seconds) or 53 ms in the real world. Using the output of the *QueryPerformanceFrequency* function and the OMNeT++ end-to-end delivery duration, a theoretical efficiency of the TSN-MIC security scheme on an AFDX network can be assumed. The theoretical performance considers the TSN-MIC operations at each AFDX component. The assumed impact of the TSN-MIC security scheme is given as 157.4 ms ($3 \cdot 26.6$ ms (LHC) + $4 \cdot 19.4$ ms (SHC)) over a simple network. This theoretical delay is 0.0339 simsec and represents a percentage in-crease of 312% above the above $10.88\text{x}10^{-6}$ simsec baseline.

The OMNeT++ model modified to create a TSN-MIC based model is then executed to determine the overhead of the TSN-MIC calculations. The outputs from the *QueryPerformanceFrequency* function are converted to simsecs and integrated into the *SendDelay()* function of the OMNeT++ model to assess the added delay. The delay caused by the verification step at the TSN Switch and destination ES is assumed to be negligible, and as such, is not considered here. The execution of the OMNeT++ TSN-MIC simulation shows that the actual end-to-end delay for a single message is $11.84 \text{x} 10^{-6}$ simsec, an increase of 8.82% above the baseline of $10.88 \text{x} 10^{-6}$ simsec. To determine if and how this delay overhead changes as the size of the messages also changes, additional averages were calculated for messages of sizes ranging from 40 bytes to 1500 bytes as shown in Tab. 3. It is observed that the overall increase in the simulation time (simsec) is much greater (2,032%) than the change in the TSN-MIC LHC processing time (111%). As the TSN-MIC SHC will always operate over the same size data, there is no change expected in the processing time, and as such, is not considered in this comparison.

Tab. 3: OMNeT++ efficiency versus TSN-MIC efficiency with increasing message size

Message Size (bytes)	TSN-MIC LHC average processing time (ms)	OMNeT++ processing time (simsec)
40	22.8	$3.84 \text{ x } 10^{-6}$
150	25.4	$12.64 \text{ x } 10^{-6}$
300	29.7	$26.64 \text{ x } 10^{-6}$
450	31.3	$36.64 \text{ x } 10^{-6}$
600	33.4	$48.64 \text{ x } 10^{-6}$
750	36.3	$60.64 \text{ x } 10^{-6}$
900	39.6	$72.64 \text{ x } 10^{-6}$
1150	44.3	$92.64 \text{ x } 10^{-6}$
1500	48.1	$119.36 \text{ x } 10^{-6}$
Percentage increase	+111%	+2,032%

Considering the actual 8.82% increase in transmission duration for an end-to-end transmission (3 LHCs and 4 SHCs) across a simple network, where it is assumed that the seven (7) calculations are equal, then per pair of TSN-MIC calculations (1 SHC and 1 LHC), the percentage delay incurred is 2.52% on average, which falls within the lower percentile of the theoretical range (1% to 35%) as given in Tab. 2. This indicates that the TSN-MIC scheme functions within the theoretical range. However, the differences in the testing environments (microcontroller versus laptop with OMNeT++) and implementation (hardware versus software) means that a true 1:1 comparison is not feasible. However, the OMNeT++ performance does indicate that the overall impact of the TSN-MIC security scheme on network performance is less significant than the impact of the frame payload size. Therefore, the integration of the TSN-MIC scheme should not negatively impact the time- and mission-critical services of the selected time-sensitive network (TSN).

5 Conclusion

The TSN-MIC efficiency based on the OMNeT++ shows an increase in the transmission time of 8.82% for each message from source End System to a destination End System, traversing a single TSN Switch. This indicates a 2.52% delay per pair of TSN-MIC calculations (1 LHC and 1 SHC). However, with the limitations of the testing environment, where more accurate solutions/environments are used, such as with comparable microcontrollers or with an FPGA, an accurate representation of the efficiency TSN-MIC security scheme can be obtained. Nevertheless, TSN-MIC demonstrates viability for I4.0/Smart manufacturing and critical infrastructure.

Bibliography

[IS16] ISO/IEC: ISO/IEC 11770-2:2016 Information Technology – Security techniques – Key management – Part 2: Mechanisms using symmetric techniques, 2016.

[IS17] ISO/IEC: ISO/IEC CD 29192-6 Information technology -- Security techniques -- Lightweight cryptography -- Part 6: Message authentication codes (MACs), 2018.

[MM14] Mouha, N., Mennink, B., Van Herrewege, A., Watanabe, D., et. al.: Chaskey: A Lightweight MAC Algorithm for Microcontrollers. Selected Areas in Cryptography -SAC 2014, Lecture Notes in Computer Science 8781, A. Joux and A. Youssef, Springer-Verlag, 2014.

[Mo15] Mouha, N.: Chaskey: a MAC Algorithm for Microcontrollers – Status Update and Proposal of Chaskey-12. https://hal.inria.fr/hal-01242648/document. [Research Report] Inria Paris Rocquencourt, 2015.

[VH12] Varga, A., and Hornig, R.: Avionics full-duplex switched Ethernet model for OMNeT++, https://github.com/omnetpp/afdx, 2017.

[WS18] Watson, V. and Sassmannshausen, J.: Time-sensitive Ethernet Technology for next Generation CPS/I4.0, GIACM Workshop I4.0, Berlin, September 24, 2018.