Enabling and Optimizing MACsec for Industrial Environments

Tim Lackorzynski, Gregor Garten, Jan Sönke Huster, Stefan Köpsell, and Hermann Härtig

***Abstract*—Industry 4.0 will revolutionize industrial automation. Yet, future smart factories will not be created from scratch. They will rather evolve from existing legacy installations. Consequently, also industrial networks will evolve and the result will be a mixture of new and legacy components. This will make new security mechanisms necessary, that are specifically designed for this industrial use case. This work proposes modifications for MACsec [1], a new security protocol for protecting communication traffic. These modifications enable MACsec to work within future industrial settings, circumventing drawbacks introduced by legacy networking technologies. Furthermore, we managed to significantly increase the performance of MACsec.**

***Index Terms*—Industry 4.0, industrial IoT, IIoT, industrial automation, industrial networks, legacy, low-latency, real-time, security, safety.**

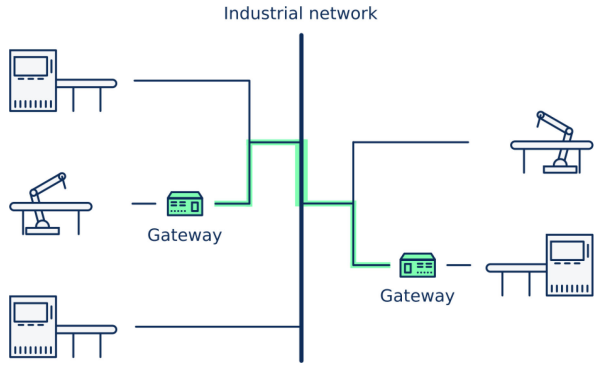


Fig. 1. Prototypical industrial network with a heterogeneous set of devices. Two gateways enable secure communication over an insecure network.

I. Introduction

NEW trends in factory automation like Industry 4.0 will introduce many innovations. New paradigms will build on new technologies that will vastly increase the amount of networking done within a factory, resulting in many new types of devices, networks, and machinery. However, most factories will not be built from the ground up as existing factories will rather be modified and updated [2]. This is due to the typically very long life cycles of industrial machines. Additionally, in many cases, these machines are never updated once deployed, as the downtime necessary for updating those machines is generally deemed too expensive and the risk of some malfunction after the update too high. This makes them insecure [3]. Furthermore, even more modern industrial applications tend to be designed without security in mind [4].

As a consequence, factory networks must become zero trust networks, where not even devices within the local network are trusted, as opposed to the legacy security model, where security mechanisms used to be only employed to separate the "good" and safe local network in its entirety from the "bad" and insecure Internet (socalled perimeter security).

One approach to increase the security of those heterogenous future industrial networks containing various new and legacy devices is to add industrial gateways [5]–[7]. These are placed in front of insecure devices and serve two purposes. One, these gateways can control all communication flowing from and to the device, practically shielding them from the rest of the network. Second, they can establish encrypted tunnels between gateways enabling secure communication of devices over an insecure network (see Fig. 1 for reference).

There is a plethora of software tools that may run on these gateways and which can be used for the purpose of encrypting data traffic. Yet, certain challenges present themselves, when applying these general purpose tools to an industrial setting. Legacy networking technology, which is typically rolled out in existing factories as well as performance requirements for latency-critical applications must be taken into account.

We identified the new encryption protocol MACsec [1] as a very good starting point to face these challenges. Our contributions are twofold. First, we investigate modifications that would enable MACsec to secure the communication within legacy networks and, therefore, make it possible for MACsec to run on industrial gateways and second, we improve MACsecs

Manuscript received February 7, 2020; revised April 29, 2020, June 26, 2020, and October 6, 2020; accepted November 8, 2020. Date of publication November 26, 2020; date of current version July 26, 2021. This work was supported by the SAB (Development Bank of Saxony) under frameworks from both the ERDF (European Regional Development Fund) as well as the ESF (European Social Fund) and by public funding of the state of Saxony/Germany. Paper no. TII-20-0618. *(Corresponding author: Tim Lackorzynski.)*

Tim Lackorzynski, Gregor Garten, and Hermann Härtig are with the Faculty of Computer Science, Technical University of Dresden, 01069 Dresden, Germany (e-mail: Tim.Lackorzynski@mailbox.tu-dresden.de; gregor.garten@tu-dresden.de; hermann.haertig@tu-dresden.de).

Jan Sönke Huster is with the Technical University of Darmstadt, 64289 Darmstadt, Germany (e-mail: jan_soenke.huster@tu-dresden.de).

Stefan Köpsell is with the Barkhausen Institute, 01187 Dresden, Germany (e-mail: stefan.koepsell@barkhauseninstitut.org).

Color versions of one or more of the figures in this article are available at https://doi.org/10.1109/TII.2020.3040966.

Digital Object Identifier 10.1109/TII.2020.3040966

efficiency especially with industrial use cases in mind. Highest possible efficiency translating into low overheads for latency and throughput is an important prerequisite for any additional security layer to be considered in the first place.

The rest of the article is organized as follows. Section II will provide technical background and motivate our proposed modifications in detail. Related work will be discussed in Section III. Section IV will present a modification to MACsec that enables it to be applied in scenarios, where the legacy networking technology Fast Ethernet (IEEE 802.3u [8]) is deployed. Section V investigates on how to improve the performance of MACsec especially in industrial use cases. Finally, Section VI concludes this article.

II. Background

MACsec is a relatively new security protocol. It was standardized as IEEE 802.1AE in 2006 [9]. An implementation in the Linux kernel was introduced in 2016, making it widely available for the first time [1]. It offers encryption on open systems interconnection model (OSI model) layer 2,[1] meaning it protects whole Ethernet frames by encrypting the payload and adding additional headers. VPN solutions, like IPsec[2] or the newer Wireguard [10], operate on OSI model layer 3, meaning they protect IP packets. Therefore, they can handle layer 3 traffic only. However, this assumption cannot be guaranteed to hold within the industrial environment, especially when considering certain industrial Ethernet solutions (e.g., IEC 61158 EtherCat), which work on layer 2. Supporting these and other even more legacy installations [11] makes the use of layer 2 encryption protocols a necessary tool to secure industrial networks. Furthermore, previous research has shown, that MACsec ranks among the highest performing encryption protocols [12], making it most suitable for performance-critical environments, where security is nonetheless necessary.

Additionally, Fast Ethernet is still the prevalent networking technology in industrial settings. It was introduced in 1995 and, because of the very long life cycles of factory equipment, it will remain the backbone networking technology for many years. Furthermore, cabling is not easily replaced in factories and industrial installations. It is often an integral part of a building, being cast in foam insulation as cable ducts need to be sealed for fire safety reasons. Also, replacing cabling that connects multiple buildings on a factory ground would typically be very expensive. Moreover, Fast Ethernet is the only allowed transport medium for certain industrial communication protocols like Profinet.[3]

Fast Ethernet only allows for frames of a maximal frame size of 1518 B. After subtracting Ethernet headers, this results in a frame being able to transport at maximum 1500 B of payload in a single transaction. This value is called the maximum transmission unit (MTU). If MACsec is applied to an Ethernet frame that is already at maximum size, the addition of the necessary MACsec headers would make the resulting frame exceed the MTU. This frame would then simply be dropped by networking devices (network cards or interfaces and switches) as it would exceed the specifications of Fast Ethernet. Certain services and applications that require transmissions of large MTUs would fail silently, meaning without emitting error or failure codes and messages as they typically are not prepared to deal with transmission errors. These services and applications include for e.g., software updates, supervisory video streams, or transport layer security (TLS) handshakes necessary for remote connections for maintenance purposes.

Jumbo frames, frames with a vastly increased MTU, were only introduced with gigabit Ethernet and cannot be assumed to be available in industrial environments. The path MTU discovery technique was invented as a tool to detect unknown MTUs on the transmission path of IPv4 connections. Yet, as it works on layer 3, it cannot be expected to be available either.

These considerations make it necessary to fragment frames, i.e., to split too large frames into smaller ones, which fit a particular MTU. However, according to the OSI layering model, fragmentation is a feature provided by layer 3. Yet, we cannot assume layer 3 networking to be present and hence we cannot assume fragmentation to be available. Therefore, in order to be able to use MACsec in industrial environments, it is necessary to implement a fragmentation scheme directly into MACsec. Section IV presents and discusses our approach.

Furthermore, MACsec is specified to use a single cryptographic algorithm, AES[4] in Galois Counter Mode (AES-GCM). This is a socalled authenticated encryption with associated data algorithm (AEAD), which means that it encrypts and authenticates data in one step. This is generally a good choice for generic IT environments, as modern general-purpose CPUs provide AES hardware acceleration. Yet, this assumption does not hold for the majority of CPUs found in embedded systems for industrial environments. Research has shown, that on platforms, which do not possess AES hardware acceleration features, other cryptographic algorithms are faster [12].

As there is active research going on in the direction of efficient and high-performance cryptographic algorithms, we investigated a set of promising ciphers that could improve the performance of MACsec in industrial environments. Section V presents our findings.

III. Related Work

There exist many software solutions that can in principle encrypt traffic on layer 2. VPN software solutions like OpenVPN or IPsec offer socalled tunneling modes, where Ethernet frames are encrypted and transported over layer 3 networks. Yet, these are generally slower compared to MACsec and additionally have certain security-related drawbacks [12]. As already mentioned in the previous section, layer 3 networking cannot be assumed to be available in industrial networks, anyway. The MACsec Linux kernel module is the first and freely available implementation of a genuine, well integrated and fast pure layer 2 encryption

[1] According to the OSI layering model, standardized as ISO 7498.
[2] [Online]. Available: https://www.rfc-editor.org/rfc/rfc4301.txt
[3] [Online]. Available: https://www.profibus.com/technology/profinet/
[4] AES—Advanced encryption standard

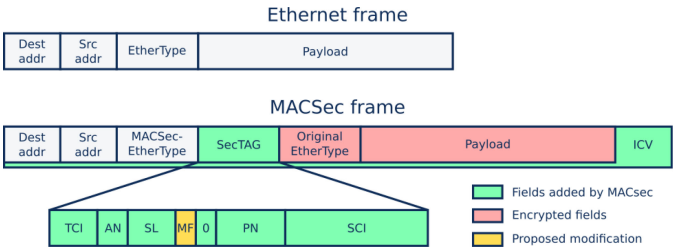


Fig. 2. Structure of an Ethernet and a MACsec frame. For clarity, irrelevant fields were omitted.

scheme, making it a de facto standard tool. It is perfectly suited for our use case.

VLAN could satisfy some of the proposed requirements [13]. It offers separation of network flows by tagging frames with an ID. As VLANs are standardized as IEEE 802.1Q, even unaware networking equipment would be able to process these frames. Yet, traffic would still be unencrypted and be sent in plain text.

IEEE 802.3br is an amendment to the Ethernet standard 802.3 and introduces a MAC Merge sublayer, which allows fragmentation of lower priority frames to support the timely delivery of express traffic [14]. Fixed time slots are allotted for different priorities. Low priority frames get preempted, when their time slot ends. This means that those frames are fragmented in a way, that utilizes the remainder of their time slot. Once the high priority time slot is finished, the remainder of the low priority frame is sent. Fragments are buffered at the receiver side and eventually reassembled. The fragmentation mechanism operates on timing schedules and not MTU sizes. It also cannot be instrumented by higher layers (such as MACsec) for such a purpose. Finally, it operates by modifying the preamble of the frame so that the receiving side can distinguish fragments from complete frames. This approach reduces this fragmentation scheme to operate on a link-by-link basis, as each intermediate piece of Ethernet equipment would buffer those fragments. It does not offer end-to-end fragmentation, which is our intended use case. All these properties make this scheme unsuitable for our needs.

IV. MACsec Fragmentation

This section presents our implementation of a fragmentation scheme for MACsec. The design of our approach is introduced and discussed in Section IV-A. We conducted a series of tests to evaluate the performance of our implementation. The testing methodology is presented in Section IV-B and the results are shown in Section IV-C.

A. Design

The structure of a MACsec frame is depicted in Fig. 2. Ordinary Ethernet frames consist of fields for source and destination address, EtherType (indicates the protocol of the payload), and the payload itself. MACsec adds two fields as its own headers. One is the security TAG (SecTAG), which contains fields and flags that are necessary for the handling of each frame. In the following, these will only be further discussed, if they are important to understand our fragmentation scheme. Detailed

Fig. 3. Fragmentation and reassembly of an incoming frame that exceeds the MTU of a MACsec-protected transport channel.

descriptions can be found in the IEEE 802.1AE (MACsec) standard [9]. Included in the SecTAG are two bits, which are not (yet) assigned by the MACsec standard. We utilized one of those bits for our fragmentation solution. We called it the more fragments (MF) flag. The second field added by MACsec is an integrity check value (ICV) that is formed over the whole frame and then appended to it.

Fig. 3 illustrates our fragmentation scheme. It starts by first checking, whether an incoming frame needs to be fragmented. This is done by comparing the payload size of the incoming frame to the MTU of the transport channel. If the frame exceeds the channel MTU, the payload is split accordingly to fit the smaller MTU. When splitting, the minimal frame size for Ethernet frames is respected. New Ethernet frames are created for each payload fragment, copying the headers from the initial incoming frame. Then, each of these new frames is individually encrypted by MACsec. Additionally during this process, the MF flag is set in the header of every fragment except for the last one. This makes it possible for the receiver to register, which frames should be reassembled. It decrypts all incoming frames and if the MF flag is set, waits for MF. When the last fragment is received (no MF flag), it reassembles the fragments into a single frame. Order is kept via the packet number (PN) fields of the headers.

Our modification of MACsec does not reduce its security, if we assume standard MACsec to already work correctly. Our approach only deviates from the original MACsec protocol in one way. During the creation of the MACsec header, we modify one bit within it (the MF flag). The whole header subsequently still gets integrity-protected by the ICV, meaning that also the MF flag is protected. Therefore, it cannot be spoofed by an attacker to make a recipient reassemble fragments in a wrong way.

Our approach also modifies the input, that is given to MACsec before encryption (fragmentation) as well as the output that is generated by MACsec after decryption (reassembly). Yet, this has no influence on the security of MACsec, as no security-related functionality is modified.

B. Implementation and Test Methodology

We implemented our approach by modifying the existing MACsec Linux kernel module.

To understand the behavior of our fragmentation solution, we tested multiple scenarios in which frames of different sizes had to be transported, resulting in different amounts of fragments each time. To establish a baseline, in the first scenario, we chose the largest size possible of the incoming frames so that still no fragmentation would happen on the transport channel. In the second scenario, we slightly increased the size of the incoming frames, so that fragmentation would happen. As a result, a big and a small fragment would be sent over the transport channel. In the last scenario, we chose very large incoming

TABLE I
CONFIGURED MTU SIZES FOR EACH SCENARIO. 1468 BYTES ARE THE EFFECTIVE MTU OVER FAST ETHERNET, WHEN MACSEC OVERHEADS ARE FACTORED IN

#	Scenario	Incoming payload size	Transport MTU
1	No Fragmentation	1468 Byte	1468 Byte
2	Small and Big Fragment	1500 Byte	1468 Byte
3	Two Big Fragments	2936 Byte	1468 Byte

frames that would result in two big fragments. The scenarios were implemented by injecting incoming frames with different payload sizes (denoted as "incoming payload size" in Fig. 3). Table I shows the respective payload and transport channel MTU sizes for each scenario.

As there is no other layer 2 encryption scheme available that can also fragment Ethernet frames, we compared our approach to a standard MACsec tunnel with a layer 3 fragmentation protocol running on top. We used the layer 2 tunneling protocol (L2TP) implementation from the iproute2 tool collection for that purpose.[5] Additionally, we measured the unmodified standard MACsec in scenario one to assess the efficiency of our implementation.

The test setting used to assess the performance of our implementation consisted of two nodes connected via Ethernet cable. This simple test setting was chosen, because we wanted to focus on the performance overhead our solution added to the transmission process compared to the standard solution. Therefore, we chose a setting with reduced complexity so that side effects from cross flows or intermediate networking equipment could be omitted as much as possible. The nodes' characteristics can be found in Table III under 1. We chose a platform with a powerful CPU and properly integrated Ethernet hardware in order to eliminate possible bottlenecks from these directions. This platform was equipped with Gigabit Ethernet hardware. Yet, this should not pose any problems for the validity of the experiments, as the for these experiments important characteristic, the MTU, is identical between Fast and Gigabit Ethernet (1518 bytes). Gigabit Ethernet is in principle capable of transmitting bigger frames, socalled jumbo frames. Yet, this has to be specifically configured and this was not done throughout the experiments.

The obvious choice to mimic legacy Fast Ethernet devices in a factory scenario would have been the Raspberry Pi 3 platform, which we also used (among others) in the evaluation in Section V. However, it is equipped with a weak CPU and its networking interface is internally connected via USB. This slows the whole networking stack down by a big margin and also increases variance considerably. The two approaches we wanted to compare (MACsec with fragmentation and MACsec with an L2TP tunnel on top) use the networking stack to a different extent and together with a strained CPU, this would lead to measurement results that could not be attributed clearly to the difference of the approaches, but rather to their usage of this Ethernet-via-USB bottleneck configuration under a busy CPU. Therefore, we chose a platform without these drawbacks.

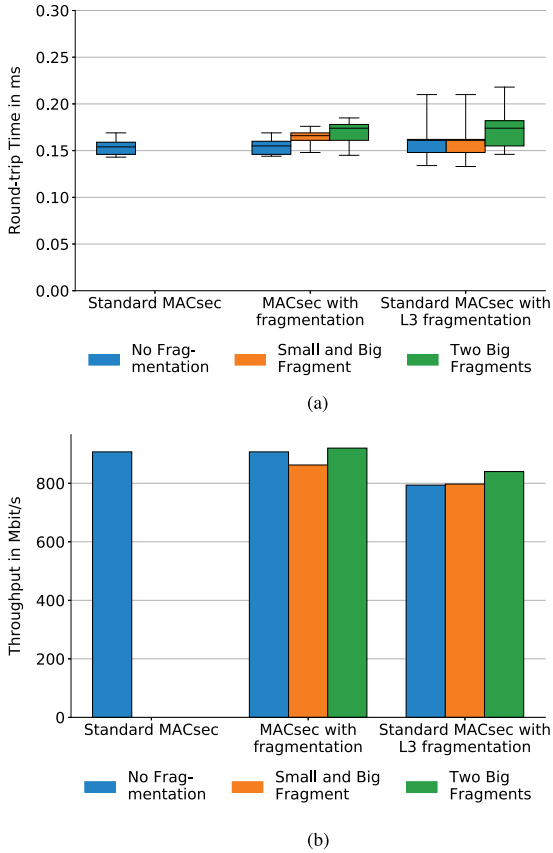


Fig. 4. Performance of our implementation of MACsec using fragmentation compared to standard MACsec only and while using layer 3 fragmentation. (a) Round-trip time measurements. (b) Throughput measurements.

The performance measures tested for were throughput and latency. We used iperf3[6] for throughput and ping round-trip times for latency measurements. While iperf3 automatically selected the biggest possible frame size, ping was configured for each scenario to produce frames with sizes according to Table I. The iperf3 experiments were conducted in TCP mode. For each scenario, 1000 iperf3 runs (10 s each) were conducted, while 50 000 ping round-trip times were measured.

C. Evaluation

Fig. 4(a) shows the results for the round-trip time measurements of the experiments described abovementioned. Each bar shows minimal, mean, and maximum values, as well as 25% and 75% quantiles.

Our approach shows expected behavior. In case of no fragmentation, it shows identical behavior compared to the standard implementation of MACsec. This shows, that our implementation is as efficient as the standard and does not create unnecessary overheads when no fragmentation is performed. The second scenario shows an increased mean round-trip time of 7% compared to the first scenario. The relatively big increase of our implementation stems from the fact, that fragmentation

[5][Online]. Available: https://man7.org/linux/man-pages/man8/ip-l2tp.8.html

[6][Online]. Available: https://software.es.net/iperf/

takes place and one frame is transformed into two. This means that an additional frame needs to be queued and sent, resulting in an increase of the total amount of bytes that need to be transmitted over the transport channel during one round-trip time measurement. The third scenario shows yet a further increase as even more payload is transmitted per measurement in relation to scenario two. Yet, since also only two frames are transmitted, the increase is only 3% from scenario two.

The standard approach with layer 3 fragmentation overall shows comparable behavior. Yet, while the mean round-trip times for all scenarios are roughly the same compared to our approach, the variance is much higher. This stems from the fact, that the layer 3 fragmentation scheme (L2TP) is its own protocol and works with its own queues and networking code. This results in an extra step, where frames are being stored and read from memory and where the operating system might preempt the transmission process. Additionally, L2TP adds another header. This results in an additional fragment in scenarios one and three, as the respective MTUs in those scenarios were chosen so that the channel MTU would be maxed.

Fig. 4(b) shows the results of the throughput measurements of the same experiments. The bars show mean values over all experimental runs for each scenario as described abovementioned. Extreme values and quantiles are not shown, as the variance within the results was comparatively low and not significant for the remainder of the considerations.

Again, our approach shows identical behavior compared to the standard MACsec implementation when no fragmentation takes place, as it does not add any overheads. The second scenario shows a decrease in performance of 50 MBit/s. Here, the second fragment is small but still has its own fully sized header. For this small frame, the header to payload ratio is very bad and effectively reduces the overall throughput by that margin. In the third scenario, two big frames are being transmitted over the transport channel. The performance increases by 60 Mbit/s compared to the second scenario, achieving roughly the same performance as in the first scenario. The reason being the better ratio between header and payload, as was described abovementioned.

Compared to our approach, standard MACsec with layer 3 fragmentation always performs worse. As described abovementioned, L2TP requires its own header, which reduces the achievable throughput over the transport channel. This additional header already makes fragmentation happen in scenario one, explaining the small difference to scenario two. The throughput then increases slightly in scenario three although a third fragment is sent, because of the dynamics explained abovementioned.

In summary, the performance results show, that our proposed modifications are efficient and do not introduce additional amounts of overhead. Furthermore, compared to a standard solution, our approach can reduce the variance of latency and increase the throughput by some margin.

V. MACsec Performance Improvements

This section presents our findings in improving MACsec performance by replacing the default cipher AES-GCM with ciphers, that may be more suited to industrial use cases. Those ciphers are presented and their selection is motivated in Section V-A. Section V-B will introduce the setting used to test those ciphers and Section V-C reports on the results.

TABLE II
SET OF INVESTIGATED CIPHERS. AES-NI STANDS FOR ADVANCED ENCRYPTION STANDARD NEW INSTRUCTIONS

#	Cipher	Key Length	Relevant Optimizations
1	AES-GCM (default)	128 bit	AES-NI
2	AEGIS	128 bit	AES-NI
3	MORUS	128 bit	-
4	ChaCha20-Poly1305	256 bit	-

A. Cipher Selection

There is much research going on in the direction of efficient high-performance cryptography. The competition for authenticated encryption: Security, applicability, and robustness competition tries to steer this research [15]. It calls for AEAD algorithms applicable to one of three different use cases. The second use case "high performance applications" is concerned with efficiency for 32 and 64 b CPU architectures and corresponds best to our use case. For example, industrial gateways fall into that category. From the three finalists of this second use case, we chose two. The third finalist was at the time of conducting this study still partly patented and hence not considered further. We chose AEGIS, which uses the AES round function and is therefore compatible to AES hardware acceleration features [16]. It tries to be more efficient compared to standard AES, while retaining a comparative security level. We also chose MORUS, which only uses simple mathematical operations (AND and XOR) for encryption, as these can be efficiently implemented independently of certain features of the underlying CPU architecture [17].

Additionally, we selected the ChaCha20-Poly1305 AEAD system, which is standardized in RFC 7539.[7] It is not related to AES and designed to be fast and efficient independently of the hardware platform it runs on. It has shown to be considerably faster on platforms, where no AES hardware acceleration features are available [12]. Table II lists the selected ciphers. We also measured the default MACsec cipher AES-GCM for comparison.

B. Methodology and Implementation

The test setting consisted of two nodes, which were connected via Ethernet cable. All ciphers were tested on four different hardware platforms. These are listed in Table III. Additionally, an unencrypted plain Ethernet connection between the nodes was measured to gain an upper bound of what is theoretically possible on each platform (denoted as 'baseline" in the remainder of this work). The performance parameters tested for were latency and throughput. The latency was measured for different frame sizes,

[7][Online]. Available: https://www.rfc-editor.org/rfc/rfc7539.txt

TABLE III
HARDWARE PLATFORMS USED FOR EXPERIMENTS

#	Platform	Processor	RAM	Network Interfaces	Operating System	Class
1	HP ProDesk	Intel Core i5-4590 Quad-Core @ 3.3 GHz	16 GB	Gigabit Ethernet	Linux 4.16.0/Debian Buster	Desktop/Generic (AES-NI available)
2	HP ProLiant MicroServer Gen7	AMD Athlon II Neo N36L Dual-Core @ 1.3 GHz	1 GB	Gigabit Ethernet	Linux 4.16.0/Debian Buster	Embedded system
3	Raspberry Pi 3	ARM Cortex-A53 Quad-Core @ 1.4 GHz	1 GB	Gigabit Ethernet (via adapter)	Linux 4.14.32/Raspbian Stretch	Embedded system
4	Raspberry Pi 4	ARM Cortex-A72 Quad-Core @ 1.5 GHz	4 GB	Gigabit Ethernet	Linux 5.0.21/Raspbian Buster	Embedded system

TABLE IV
FRAME SIZES AND WEIGHTS USED TO CALCULATE THE WEIGHTED LATENCY

Frame size (bytes)	128	256	512	1024	1400	1518
Weight	0.5	0.2	0.1	0.1	0.05	0.05

as it typically varies between small and big frames. The frame sizes were chosen within the boundaries of standard Ethernet, so that no fragmentation would take place.

All of the selected ciphers were available within the Linux Crypto API. MACsec had to be modified slightly to work with each different cipher as the length of the ICV field had to be adapted to the respective block size.

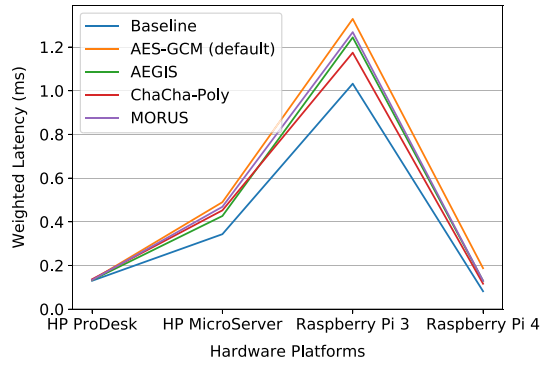
We used iperf3 for throughput and ping for latency measurements. Iperf3 was used in TCP mode. For each experiment, 100 iperf3 runs (10 s duration each) were conducted, while 50 000 ping round-trip times were measured for each frame size.

C. Evaluation

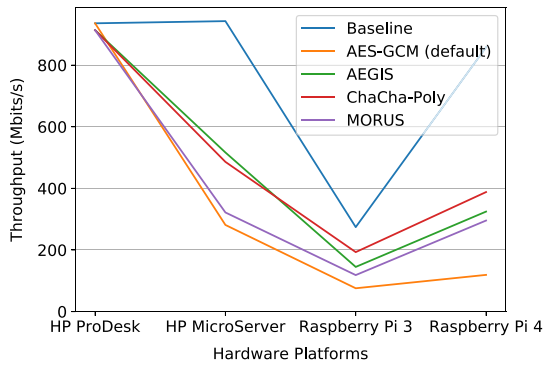
The ciphers were tested for latency and throughput. Fig. 5(a) shows a summary of the latency measurements for each cipher on all tested platforms. To create this compact view of the results, the measurement series for each cipher (on each platform) consisting of values for different frame sizes had to be condensed into one value (the *weighted latency*). This was done by calculating a weighted arithmetic mean over the values for each frame size. These values for each frame size were themselves formed by creating the mean over 50 000 individual ping measurements. We increased the weights of smaller frame sizes as this reflected industrial traffic patterns better than giving equal weight to each frame size. Table IV lists the measured frame sizes and assigned weights. The detailed results for each platform including the measurements for each frame size can be found in Fig. 6.

The results for latency show no clear picture. No cipher is consistently outperforming the others. Yet, they generally manage to reduce the latency and bring it closer to the baseline.

The results on the first platform showed only minimal divergence. Fig. 6(a) gives more detail. The encryption step was clearly not the bottleneck, as the difference between the ciphers and the baseline is minimal. In this case, the CPU was powerful enough to always saturate the networking interface independently of the cipher. The small differences probably stem



(a)



(b)

Fig. 5. Performance comparison of ciphers over all platforms. (a) Latency measurements. (b) Throughput measurements.

from each cipher's individual implementation on this platform. The AES-based ciphers profit from the available AES hardware acceleration.

The second platform showed more divergence in results. All ciphers performed slightly better compared to the default cipher. Yet, no cipher clearly outperformed the others, when looking at the results for each frame size [see Fig. 6(b)]. Nonetheless, AEGIS showed the best overall performance. This is insofar surprising, as AEGIS' argument for efficiency rests on its more efficient use of AES hardware primitives compared to the standard AES-GCM. Yet, these hardware acceleration features were not available on this platform and from a theoretical standpoint, ChaCha-Poly or MORUS should have performed better. On the

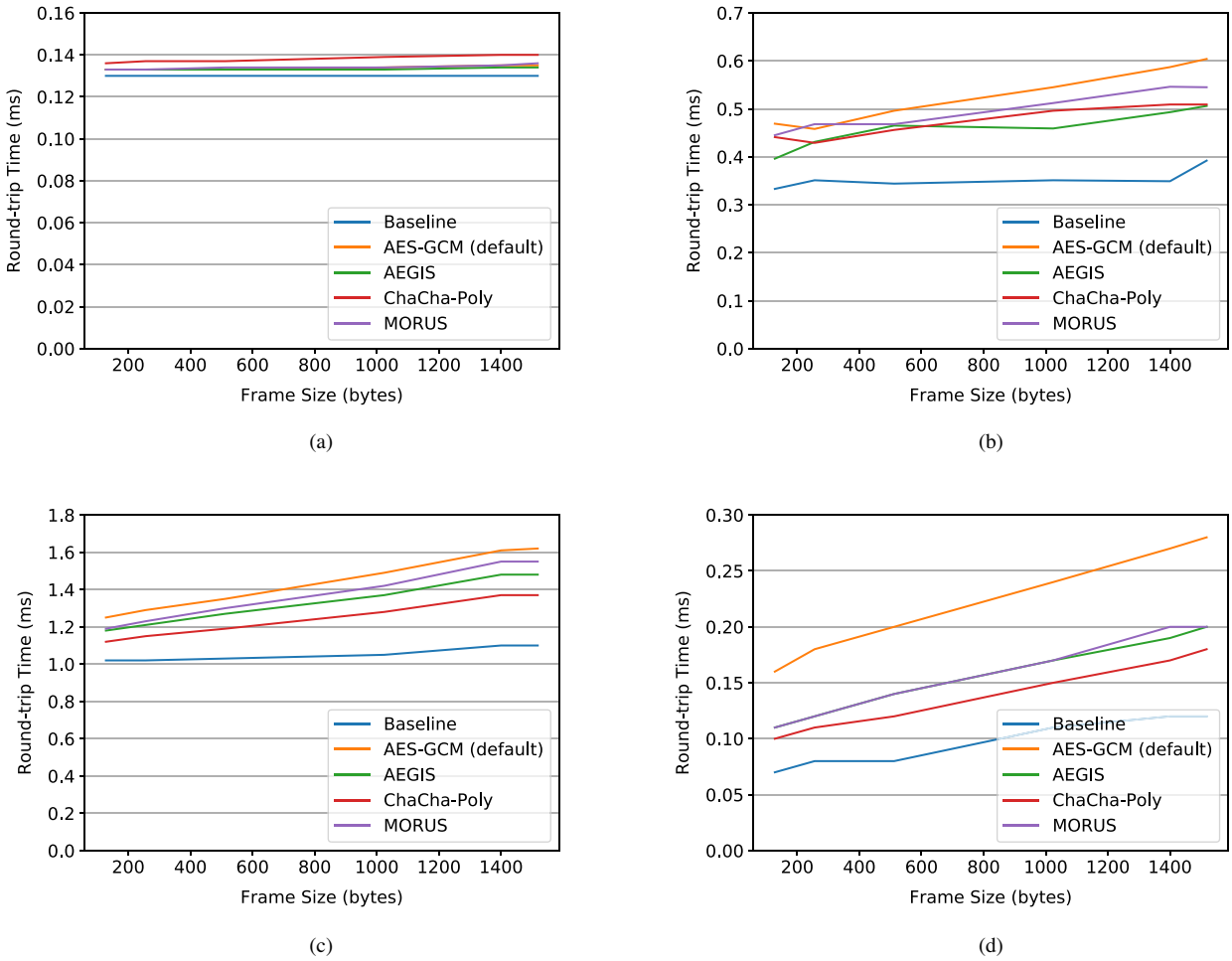


Fig. 6. Latency measurements of tested ciphers for different frame sizes for each platform. (a) HP ProDesk. (b) HP ProLiant MicroServer Gen7. (c) Raspberry Pi 3. (d) Raspberry Pi 4.

other hand, we cannot rule out implementation issues on this platform, as it is comparatively old and the cipher implementations may not be optimized for this CPU architecture.

The results of the third and the fourth platform are similar, as the latter is an update of the former with a faster CPU and a better networking interface. Both show consistent behavior over different frame sizes [see Fig. 6(c) and (d)], giving higher confidence to the results, compared to the previous platform. AEGIS and MORUS showed increased performance compared to the default cipher, while ChaCha-Poly achieved the best improvements. The fourth platform showed a bigger relative distance between the performances of the baseline and the default cipher compared to the other platforms, giving the most space for improving the performance with selecting a different cipher. In this case, ChaCha-Poly managed to decrease the round-trip time for all frame sizes by 40% compared to the standard implementation.

Fig. 5(b) shows the combined results of the throughput measurements for each cipher on all platforms. No cipher performs consistently better than the others and the results generally mirror the latency results discussed abovementioned. The rankings between the ciphers stays the same. Only the relative gains, when compared to the default cipher, are bigger.

Again, the performance of the first platform cannot be improved by one of the selected ciphers. On the contrary, they reduce the throughput by 2.5%, probably due to deficiencies in our prototypical implementation.

The second platform shows vast divergence between the default cipher and the baseline, hinting that the CPU of this platform is clearly lacking necessary computing power to saturate its own networking interface when also encrypting the traffic. This opens up big potential for more efficient ciphers and consequently, AEGIS and ChaCha-Poly manage to almost double the achieved throughput (184% and 173%, respectively), compared to the default.

The third and the fourth platform again showed similar behavior only differing in the relative distance of the results because of their different CPUs. All ciphers significantly improved the performance. ChaCha-Poly managed a throughput of 255% and 326%, respectively, compared to the default.

The experiments showed that resulting improvements depend on the underlying hardware platform. Significant throughput improvements could be achieved, while improvements for latency were, in absolute terms, small. Especially, the results from the second and the fourth platform showed, that when the computing power of the CPU in relation to the attached networking interface

is low, much can be gained from choosing a more efficient cipher. AEGIS and ChaCha-Poly achieved the biggest performance increases and AEGIS even worked efficiently in a scenario, it was not primarily designed for. Therefore, it may even be considered for platforms without AES hardware acceleration features.

VI. Conclusion

MACsec was a new standard protocol for the encryption of Ethernet frames. This article investigated necessary steps to enabled and improved it for industrial use cases.

We identified the need for a fragmentation scheme within MACsec due to certain particularities of industrial networks stemming from the frequent use of legacy networking equipment. Our proposed scheme worked efficiently and did not incur overheads when fragmenting and even slightly improved performance compared to a standard solution.

Additionally, this article identified two highly efficient cipher algorithms that can significantly improve the performance of MACsec on embedded platforms typical for industrial use cases. Especially ChaCha-Poly showed vast potential on platforms without AES hardware acceleration features. As it was also by default included in the Linux kernel, this cipher should be considered more in comparable used cases in the future.

Further research should investigate related use cases, where the proposed fragmentation scheme might yield additional benefits. While the two ciphers identified, AEGIS and ChaCha-Poly, turned out to considerably improved the performance of MACsec, it would be beneficial to investigate further ciphers for even higher improvements.

References

[1] S. Dubroca, "MACsec: Encryption for the wired LAN," in *netdev 1.1*. Red Hat, Feb. 2016. [Online]. Available: https://www.youtube.com/watch?v=LZTxmOMGpwc

[2] K. Stouffer, S. Lightman, V. Pillitteri, M. Abrams, and A. Hahn, "Guide to industrial control systems (ICS) security," *Nat. Inst. Standards Technol.*, Gaithersburg, MD, USA, May 2015. [Online]. Available: http://dx.doi.org/10.6028/NIST.SP.800-82r2

[3] A. Greenberg, "The untold story of NotPetya, the most devastating cyberattack in history," Wired.com, Sep. 2018. [Online]. Available: https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-code-crashed-the-world/

[4] D. Quarta, M. Pogliani, M. Polino, F. Maggi, A. M. Zanchettin, and S. Zanero, "An experimental security analysis of an industrial robot controller," in *Proc. IEEE Symp. Secur. Privacy*, May 2017, pp. 268–286.

[5] A. Bluschke *et al.*, "Fastvpn— secure and flexible networking for industry 4.0," in *Proc. 12th ITG Conf. Broadband Coverage*, Apr. 2018, pp. 1–8.

[6] Y. Huang, Z. Zhang, and P. Zhu, "The design of an industrial remote control network gateway based on p2p VPN," in *Proc. 4th Int. Conf. Intell. Hum. Mach. Syst. Cybern.*, Aug. 2012, pp. 140–143.

[7] T. Yamada and T. Maruyama, "Study on a security framework for a plant level network," in *Proc. SICE-ICASE Int. Joint Conf.*, Oct. 2006, pp. 1063–1066.

[8] *IEEE Standards for Local and Metropolitan Area Networks*, IEEE Std 802.3u-1995 (Supplement to ISO/IEC 8802-3: 1993; ANSI/IEEE Std 802.3, 1993 Edition), Oct. 1995.

[9] *IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Security*, IEEE Std 802.1AE-2006, 2006.

[10] J. A. Donenfeld, "WireGuard: Next generation kernel network tunnel," 2018. [Online]. Available: http:\\www.wireguard.com.

[11] *ICS Security Compendium, Version 1.23*, Federal Office for Information Security (BSI), Germany, 2013. [Online]. Available: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/ICS/ICS-Security_compendium.pdf

[12] T. Lackorzynski, S. Köpsell, and T. Strufe, "A comparative study on virtual private networks for future industrial communication systems," in *Proc. 15th IEEE Int. Workshop Factory Commun. Syst.*, May 2019, pp. 28–35.

[13] *IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks*, IEEE Std 802.1Q-2014 (Revision of IEEE Std 802.1Q-2011), Dec. 2014.

[14] *IEEE Standard for Ethernet Amendment 5: Specification and Management Parameters for Interspersing Express Traffic*, IEEE Std 802.3br-2016 (Amendment to IEEE Std 802.3-2015 as amended by IEEE Std 802.3bw-2015, IEEE Std 802.3by-2016, IEEE Std 802.3bq-2016, and IEEE Std 802.3bp-2016), 2016.

[15] "Caesar: Competition for authenticated encryption: Security, applicability, and robustness," Feb. 2019. [Online]. Available: https://competitions.cr.yp.to/caesar.html

[16] H. Wu and B. Preneel, "AEGIS: A fast authenticated encryption algorithm," *Selected Areas in Cryptography–SAC 2013*, vol. 8282, pp. 185–201, May 2014.

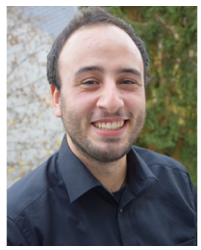
[17] H. Wu and T. Huang, *The Authenticated Cipher MORUS (v2)*, Nanyang Technological Univ., Singapore, Sep. 2016. [Online]. Available: https://competitions.cr.yp.to/round3/morusv2.pdf

Tim Lackorzynski received the Diploma degree in computer science from TU Dresden, Dresden, Germany, in 2013, where he is currently working toward the Ph.D. degree in industrial networks with the Privacy and Security group.

His research interests include security for distributed systems and industrial communication.

Gregor Garten received the B.A. degree in lightweight ciphers for embedded systems from TU Dresden, Dresden, Germany, in 2018, where he is currently working toward the master's degree in privacy-preserving smart metering.

His research interests include practical security mechanisms and performance of encryption algorithms.



Jan Sönke Huster received the B.Sc. degree in enabling transparency for layer 2 encryption protocols from TU Dresden, Dresden, Germany, in 2018, where he is currently working toward the master's degree in computer science.

His current research interests include network security, privacy and anonymity technologies as well as their impact on society.

Stefan Köpsell received his Ph.D. degree in computer science from TU Dresden, in 2010.

He has been working in the field of data security and privacy in and by distributed systems for more than 20 years. He joined the Barkhausen Institute in January 2019 where he leads the Secure and Privacy-Respecting Data Processing group. Additionally, he is working at the Chair of Privacy and Data Security at the TU Dresden, which he currently leads as an Acting Professor. His current research interests include IoT-security.

Hermann Härtig received the Ph.D. degree from the University of Karlsruhe.

He was a Professor and Head of the Operating Systems group, TU Dresden from 1994 to 2019. He currently holds the position of Senior Professor with the same group. His research interests include real-time systems, μ-microkernel-based operating systems and security in operating systems.