Cryptographic Protection of Cyclic Real-Time Communication in Ethernet-Based Fieldbuses: How Much Hardware is Required?

Matthias Skuballa, Andreas Walz, Heiko Bühler, Axel Sikora
Institute of Reliable Embedded Systems and Communication Electronics (ivESK)
Offenburg University of Applied Sciences
Badstrasse 24, 77652 Offenburg, Germany
{andreas.walz, heiko.buehler, axel.sikora}@hs-offenburg.de

***Abstract*—It seems to be a widespread impression that the use of strong cryptography inevitably imposes a prohibitive burden on industrial communication systems, at least inasmuch as real-time requirements in cyclic fieldbus communications are concerned. AES-GCM is a leading cryptographic algorithm for authenticated encryption, which protects data against disclosure and manipulations. We study the use of both hardware and software-based implementations of AES-GCM. By simulations as well as measurements on an FPGA-based prototype setup we gain and substantiate an important insight: for devices with a 100 Mbps full-duplex link, a single low-footprint AES-GCM hardware engine can deterministically cope with the worst-case computational load, i.e., even if the device maintains a maximum number of cyclic communication relations with individual cryptographic keys. Our results show that hardware support for AES-GCM in industrial fieldbus components may actually be very lightweight.**

***Index Terms*—Security, Cryptography, Real-time Systems, Industrial Automation, Benchmark**

I. Introduction

Today, it is understood that industrial communications increasingly face the demand for strong cryptographic protection against malicious data manipulations or injections [1]. The demand also applies to fieldbus communications, in particular, if these are based on widespread standard protocols like Ethernet and TCP/IP [2, 3].

Fieldbus communication systems are traditionally designed to meet strict reliability and real-time requirements [4]. It is only since some time that security requirements were added to the list. As cryptographic message protection comes with considerable computational complexity, cryptographic protection is commonly perceived to conflict with the real-time requirements of industrial fieldbuses.

Indeed, measurements on typical platforms showed that the software-based protection (i.e., authentication with or without encryption) of messages with typical state-of-the-art symmetric cryptographic algorithms easily takes more than one millisecond [5, 6]. Having said that, for Ethernet-based fieldbuses like PROFINET, one millisecond is in fact a common cycle time. Sending and receiving cryptographically protected messages at such a frequency, therefore, seems clearly out of reach. For *programmable logic controllers* (PLCs) the situation seems to be even worse: one PLC potentially needs to serve a large number of field devices at once, and this factor directly seems to scale down the prospects of applying cryptographic protection in such scenarios. Interestingly, a cryptographic co-processor does not necessarily improve the situation, as the initialization, supply of input and retrieval of output may take longer than the cryptographic operation itself [6].

On the other hand, the existence of MACsec (the IEEE standard[1] for link-layer encryption [7]) deployments demonstrate that strong cryptographic protection is possible at link speed, e.g., within routers [8]. This observation is encouraging, but deserves a closer inspection: on MACsec links there is typically only one cryptographic key in use that tends to be updated rarely (on the order of minutes or even hours). For a PLC, the situation is different, as it might need to process a burst of short messages each with an individual key (one for each communication partner) within a single cycle time (i.e., potentially within a millisecond or less). As many cryptographic algorithms require the precalculation of key-dependent internal state, updates of the key can cause additional—potentially significant—delays [9]. The high-throughput hardware powering MACsec is therefore not automatically equally suitable for protecting the cyclic communication of a fieldbus layer.

From this, system designers are left with an unsatisfactory situation: software-only implementations of cryptography were shown to not provide the necessary performance, while the use of dedicated high-throughput hardware is not proven to do better for high key switching frequencies. It led us to believe that at least PLCs will not do without *multiple* dedicated high-performance cryptographic hardware engines to serve potentially hundreds of different communication relations (CRs) at high frequency. From discussions with others involved in industrial automation, we deduce that we were not the only ones with this belief.

We set out to shed light on the matter and strive to answer two important questions with this paper. *What are the timing requirements that a cryptographic engine needs to meet in a PLC if all its cyclic communication shall be*

[1] IEEE 802.1AE

cryptographically protected?, and *Does it really require highly specialized cryptographic hardware to meet these requirements?* We approach the two questions by first calculating timing boundaries for the necessary cryptographic computations in a PLC given the Ethernet interface's bandwidth constraints. We then relate these numbers, on the one hand, to the capabilities of a cryptographic hardware engine, whose performance figures we obtain via simulations and measurements on an FPGA prototype setup, and, on the other hand, to the performance of a corresponding software implementation on an ARM Cortex A9 dual-core processor clocked with 667 MHz.

For our study, we decided to focus on the *Advanced Encryption Standard* (AES) cipher [10] in *Galois/Counter Mode* (GCM) mode [11], referred to as AES-GCM. AES-GCM is among the most popular and most respected symmetric cipher modes for authenticated encryption [12]. It is the only cipher mode that is supported by MACsec [7].

We believe our work is a valuable extension of previous work. First, we not only measure the performance of a lightweight AES-GCM hardware engine, but we relate the obtained numbers to the particular performance requirements of PLCs. Second, we study AES-GCM as a state-of-the-art algorithm that supports both authenticated encryption and authentication-only scenarios. Other work mostly concentrated on pure authentication-only algorithms (like HMAC [13]) and nowadays considered-obsolete modes (like AES-CBC [14]).

Our paper is structured as follows. Section II provides some background on industrial fieldbus communication as well as cryptography as a means to protect such communication. Section III discusses related work. In Section IV we derive timing requirements for cryptographic computations in a PLC. In Section V we consider three viable implementation options for a cryptographic engine. In Section VI we describe our evaluation setup and present the results in Section VII. Finally, in Section VIII we summarize the results and give an outlook on further activities.

II. Background

This section provides a condensed recap of (industrial) fieldbus and cryptography basics, which are important to understand this paper.

A. Fieldbus Basics

Fieldbuses are used for distributed industrial control applications like factory or process automation [4]. There is a plethora of different Ethernet-based fieldbuses, e.g., *PROFINET*, *Modbus/TCP*, or *EtherNet/IP*, most of them being standardized in the international norms IEC 61784 [15] and 61158 [16].

Ethernet-based fieldbuses differ in several respects, but all share the capability to exchange *input and output* (IO) data between involved system entities in a cyclic manner and at high frequency [17]. Cyclic here means that producers of IO data transmit the same type of data with updated content periodically, i.e., once per *cycle time* (e.g., once per millisecond). The form and multiplicity of possible producer-consumer relations, in the following referred to as *communication relations* (CRs), differ from fieldbus to fieldbus.

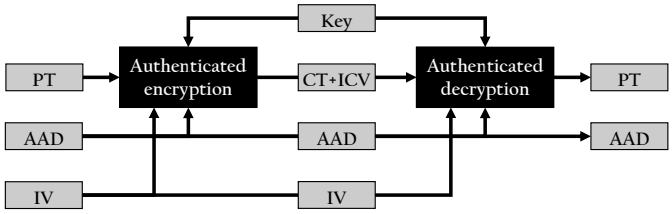

Figure 1. The *authenticated encryption with associated data* (AEAD) mode of operation illustrated. "PT" is plaintext, "CT" is ciphertext, "ICV" is interity check value, "AAD" is additional associated data, and "IV" is initialization vector. "PT", "CT", and "AAD" are variable-length data, "Key", "IV", and "ICV" are fixed-length data.

For all fieldbuses, the transmission of IO data (from producers to consumers) must succeed deterministically within certain time boundaries. Some fieldbuses target only mild time boundaries, in which case communication may be based on standard TCP/IP, whereas other fieldbuses target very tight time boundaries [18]. Fieldbuses of the latter kind often transmit IO data in layer-2 Ethernet frames directly to minimize protocol overhead.

B. Cryptography Basics

Cryptography is the major means to achieve secure communications in open networks. Symmetric cryptographic algorithms are the "*workhorse*", facilitating the protection of bulk data against disclosure (encryption) and manipulation or spoofing (authentication) using a cryptographic key shared between the legitimate communication entities. Asymmetric cryptographic algorithms are the counterpart and help to share or distribute the required symmetric keys securely. In the following, we purely focus on symmetric cryptographic algorithms.

Classically, symmetric message encryption (privacy protection) and authentication algorithms were two distinct building blocks [19]. Today, the leading category of operation modes is *authenticated encryption with associated data* (AEAD), which combines both encryption and authentication into a single primitive [20].

An AEAD primitive supports two distinct operations, which are *authenticated encryption* and *authenticated decryption* (see Fig. 1). Input to authenticated encryption is a fixed-length symmetric cryptographic key, a fixed-length unique initialization vector (IV), a variable-length bit string of *plaintext* (PT) and a variable-length bit string of *additional associated data* (AAD). It outputs a variable-length bit string of ciphertext (CT), which is the encrypted plaintext (same length but encrypted content), and a fixed-length *integrity check value* (ICV)[2]. The ICV covers both the plaintext and the additional associated data.

[2]Note that, often, the ciphertext is defined as comprising both the encrypted plaintext and the integrity check value (see, e.g., RFC 5116 [21]). Here and in the following discussions, we intentionally keep CT and ICV as separate data elements.

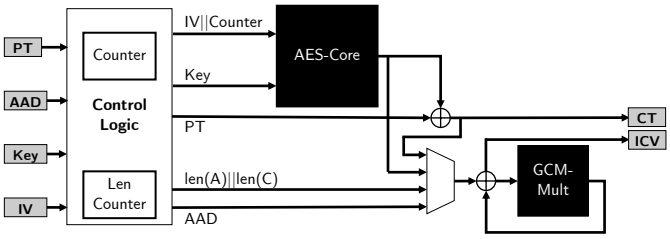

Figure 2. Simplified visualization of the AES-GCM cryptographic engine and its AEAD-interace

Input to authenticated decryption, which is the reverse operation to authenticated encryption, is the symmetric cryptographic key, the initialization vector, the ciphertext, and the additional associated data as used for authenticated encryption. It outputs the decrypted plaintext after successful verification of the ICV or an error indication if the verification failed.

An interesting feature of AEAD modes as described above is that they allow both authentication-only protection as well as authenticated encryption with the same algorithm. The choice between these two options is just governed by the input to which to-be-protected data is supplied: if given to the plaintext (PT) input, data is authenticated *and* encrypted; if given to the additional associated data (AAD) input, data is only authenticated. However, it is perfectly fine (and a quite typical case) to supply one portion of data (e.g., a message's payload) to the PT input and another portion of data (e.g., a message's header) to the AAD input.

AES-GCM refers to an AEAD mode, which uses the AES cipher [10] in *Galois/Counter Mode* (GCM) mode [11]. AES-GCM supports key sizes of 128, 192, and 256 bits. Simply put, AES-GCM performs encryption by xor-ing plaintext data with a key-dependent stream of pseudo-random data produced using the AES cipher, and authentication using a universal hash function. The ICV[3] produced by AES-GCM's authenticated encryption has 16 bytes. If AES-GCM is used in an authentication-only setting, it is sometimes also referred to as AES-GMAC.

A simplified visualization of an architecture to perform AES-GCM authenticated encryption is shown in Fig. 2.

III. Related Work

Numerous implementations and implementation optimizations of AES-GCM were proposed and studied with respect to their performance [22–25]. However, these studies did not consider the special demands of cyclic industrial fieldbus communication.

Czybik et al. [5] and Runde et al. [6] filled the gap and studied the performance of different cryptographic algorithms—including AES-GCM and AES-GMAC—for cyclic industrial fieldbus communication. Müller and Doran studied the performance of further message authentication algorithms in the context of PROFINET [26, 27]. From these studies, one can infer that cryptographic computations may easily conflict with short cycle times on a number of platforms. However, these works do not investigate the actual timing requirements that a cryptographic implementation powering a PLC must meet, and they also do not investigate the prospects of dedicated cryptographic hardware engines.

IV. Timing Requirements

Before studying the performance of a particular cryptographic algorithm, our goal is to calculate the maximum time $t_{\max}$ a cryptographic engine on a PLC has available to process either one incoming or one outgoing IO data frame. To this end, we collect all the related requirements and boundary conditions of a typical PLC in an Ethernet-based real-time fieldbus system. We perform this analysis using PROFINET [28] as the underlying technology. However, similar automation systems will lead to similar results.

We assume there is a central PLC that maintains a variable (but fixed-at-runtime) number N of *communication relations* (CRs) with different field devices. We consider a CR as a pair of reciprocal producer-consumer relations between exactly two system entities (that is, real-time data is exchanged in both directions). Our model is inspired by that of PROFINET IO, where a PROFINET *IO Controller* maintains one or more pairs of *IO CRs* to one or more PROFINET IO Devices[4]. In this spirit we also assume that cyclic IO data is exchanged within layer-2 Ethernet frames, just as PROFINET with its RTC protocol does.

We make the following assumptions:

- The PLC is equipped with a full-duplex Ethernet link with a data rate R of 100 Mbit/s, through which all its real-time communications must proceed.
- A network load η of 50% is not exceeded, that is, during one communication cycle at most half of the time is used for transmitting/receiving real-time IO data.
- The size of the IO data is between 44 and 1444 bytes.
- The size of IO data is identical for all frames.
- For each CR, there is one incoming and one outgoing frame per communication cycle.
- A single cryptographic engine is used for both incoming and outgoing frames respectively.
- Each CR is protected using a CR-specific cryptographic key.

Each message, typically IEEE 802.3-tagged Ethernet frames, sent over the line starts with an Ethernet header consisting of a preamble, start frame delimiter, source and destination MAC addresses, type field, and a VLAN tag, resulting in a 26 bytes Ethernet header. We assume that the subsequent fieldbus header (without security-related data) is two bytes long. Protected communication is enabled by adding two security-related fields: a one-byte key identifier and a four-bytes sequence counter. The length of the fieldbus header for protected frames, therefore,

[3]The specification of the GCM mode [11] refers to the integrity check value (ICV) as *authentication tag*. In some contexts, it is also called *message authentication code* (MAC). As a synonym, we use ICV throughout this paper.

[4]Note that a PROFINET IO CR is unidirectional and that, therefore, a CR as used by us maps to a *pair* of PROFINET IO CRs.

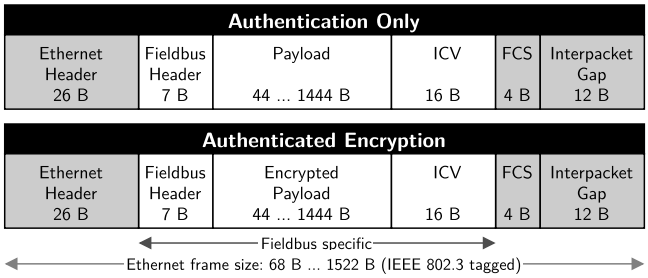

Figure 3. Frame structure of the protected frames given two operation modes: (1) *Authenticaiton Only* on top, which appends an ICV to the to-be-protected message and (2) *Authenticated Encryption* on bottom, which encrypts the payload and appends an ICV. The Fieldbus Header and the payload are the to-be-protected data.

sums to seven bytes. After that follows the actual payload (44 to 1444 bytes), an 16-bytes (128-bit) ICV, the Ethernet frame check sequence (FCS) with four bytes, and finally the interpacket gap with 12 bytes. The resulting frame layout is shown in Fig. 3. Given this frame structure, the actual Ethernet frame length is

$$\begin{aligned} l &= 26 + \Big(7 + [44 \ldots 1444] + 16\Big) + 4 + 12 \\ &= [109 \ldots 1509] \text{ bytes.} \end{aligned} \tag{1}$$

We distinguish two operating modes: *Authentication Only* and *Authenticated Encryption*. In the first mode, the fieldbus header and the payload are concatenated and given to the AAD input of AES-GCM. The resulting ICV is appended to the message payload, which is being transmitted unencrypted. In the second mode, only the fieldbus header is given to the AAD input of AES-GCM, the payload is fed into the PT input. The resulting ciphertext (which is the encrypted payload) replaces the cleartext payload in the message. Again, the ICV is appended. The respective message structures are shown in Fig. 3. The length of the inputs to AES-GCM are summarized for both minimum and maximum-length frames in Table I. Important to note here is that the resulting frames have identical lengths for both operation modes.

The maximum amount of CRs that the PLC may simultaneously maintain depends on the assumed size of the real-time payload, because the network load η and the data rate R are assumed to be fixed. The longer the payload gets, the fewer CRs are possible. The number N of CRs that can be served within a single communication cycle T given a network load η must be an integer number, which can be derived for a payload length l as follows ($\lfloor\rfloor$ denotes rounding down to an integer):

$$N = \left\lfloor \frac{\eta \cdot R \cdot T}{l} \right\rfloor \tag{2}$$

Since we assume that all frames are of the same length and that each CR uses its own key, the cryptographic engine must cope with a high key switching frequency. Furthermore, it must be fast enough to process all incoming and outgoing frames of all CRs within a single communication cycle. As a single CR results in one outgoing and one incoming frame per communication cycle, the load on the cryptographic engine is effectively doubled. Using the maximum number of CRs from Eq. (2) we get:

$$t_{\max}(l) = \frac{T}{2 \cdot N} = \frac{T}{2} \left\lfloor \frac{\eta \cdot R \cdot T}{l} \right\rfloor^{-1} \tag{3}$$

Observe that $t_{\max}(l)$ is approximately independent of the assumed cycle time T (T remains in Eq. (3) because of the rounding operation). This is plausible because one can consider outgoing or incoming cyclic IO data as a stream of independent frames. The above equation Eq. (3) relates the maximum time a cryptographic engine may take to process these frames on-the-fly to the rate with which they can pass through the network interface. From this perspective, it is irrelevant whether two chronologically-nearby frames correspond to two distinct CRs in the same communication cycle or to the same CR but in distinct communication cycles.

Using concrete numbers ($\eta = 0.5$, $R = 10^8$ bits/s, $T = 1$ ms), from Eq. (3) we obtain a maximum processing time of $t_{\max}(109 \text{ bytes}) = 8.77$ µs per frame for minimum-length frames and of $t_{\max}(1509 \text{ bytes}) = 125$ µs per frame for maximum-length frames (cf. Eq. (1)).

Table I
LENGTH (IN BYTES) OF INPUT TO THE CRYPTOGRAPHIC ENGINE FOR MINIMAL-LENGTH AND MAXIMAL-LENGTH FRAMES, SHOWN SEPARATELY FOR PT AND AAD INPUT.

	Minimum Frame Length		**Maximum Frame Length**	
Mode	PT	AAD	PT	AAD
Auth. Only	0	51	0	1451
Auth. Enc.	44	7	1444	7

V. IMPLEMENTATION OPTIONS

Given the aforementioned requirements, we consider three viable designs to implement an engine for performing cryptographic operations. These three options are: (1) All operations are computed in dedicated hardware (e.g., FPGA or ASIC), (2) involves a generic CPU (e.g., RISC) without AES specific instructions and (3) uses a generic CPU however with dedicated AES instructions to calculate the cryptographic operations.

These three options shall provide an overview of possible implementations of the core cryptographic engine. Our actual implementation is presented in Section VI.

A. HW-only

This option dedicates a piece of hardware solely to the purpose of executing cryptographic operations. As we use AES in GCM mode these operations mainly consist of substitution (e.g., lookup) and permutation (e.g., shifting and mixing) steps plus XOR operations. All these operations are easy to implement and can often be executed in a single clock cycle.

The functional structure of AES allows parallelizing these operations, for example by computing all rounds of AES in a single clock cycle by duplicating and cascading the relevant

hardware blocks. Like often in hardware designs this results in a tradeoff between throughput vs. area vs. power consumption.

B. SW-only

In contrast to the hardware-based implementation of a cryptographic engine, all the steps to compute an AES encrypted block can also be performed on a general-purpose CPU (e.g., ARMv7 or x86). A software-only approach allows introducing a cryptographic engine retroactively. Given the number of shipped devices in industrial communication systems, it is tempting to extend those devices with a software update to enable security. However, one has to consider the additional processing load. There exist manifold optimizations techniques for AES to reduce the computation load as shown in [29–31].

C. SW with HW-acceleration

Another approach to implement AES operations is by involving dedicated instruction set extensions, like AES-NI by Intel [32] or the ARMv8 cryptographic extensions by ARM. Those instructions improve the speed and security of AES instructions. Typically, these instructions perform one round of AES per instruction, which is a major performance boost compared to the SW-only solution.

VI. Implementation on Zynq-7000

In our studies, we considered the aforementioned HW-only and SW-only approach by implementing both variants on a Xilinx Zynq-7000 SoC development board. It consists of an FPGA and a processing system (PS) that contains an ARM Cortex A9 dual-core processor clocked with 667 MHz. This design choice allows us to demonstrate a proof of concept for the HW-only solution, as well as, to evaluate the performance of a SW-only solution. However, we could not pursue the third approach using SW plus HW-acceleration, because the given processor (ARMv7) does not support an AES instruction set. An overview of this system including the high level descriptions of relevant submodules is given in Fig. 4.

As the FPGA is decoupled from the PS running the application providing the to-be-protected data, an interface between those two worlds has to be established. One particular interface called CAESAR Hardware API by Homsirikamol et al. [33] has been initially developed to enable compatibility between implementations and to allow fair benchmarking of cryptographic algorithms submitted to the CAESAR competition. This competition has had the goal of comparing different authenticated encryption algorithms. The developed interface consists of three major data buses called *Public Data Input (PDI)*, *Secret Data Input (SDI)* and *Data Output (DO)*. Such an interface closes the gap between the application providing the data and the actual cryptographic engine. It can be easily implemented in FPGA-based designs or may even be used in ASIC designs. One exemplary AES-GCM implementation written in VHDL using the CAESAR Hardware API has been provided by Homsirikamol [34]. In our design, we favor reduced area consumption over high throughput and reduced power consumption.

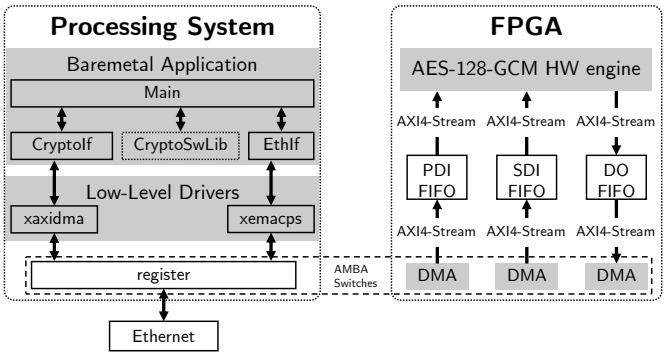

Figure 4. Simplified overview of the complete system including the FPGA and the PS (cf. [35]). The PS hosts a Main-Application that controls the cryptographic engine and handles the Ethernet communication using SW-modules denoted as CryptoIf and EthIf which interface the provided low-level drivers.

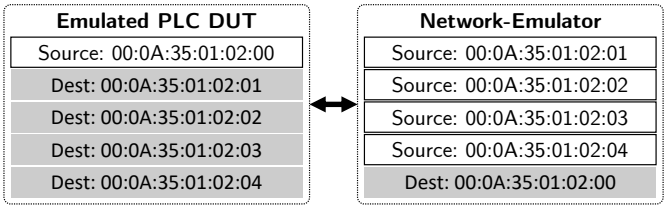

Figure 5. Prototype setup where one board acts as a PLC-DUT and the other board acts as a network emulator. The shown setup reflects a configuration with four CRs.

The FPGA implements the cryptographic engine whereas its inputs for PDI and SDI as well as its output DO are linked to AXI4-Stream Data FIFOs. These FIFOs in turn are connected to the processing system using mapped memory through high-speed DMA channels. A simple bare metal application, running on the first CPU only, controls the cryptographic engine and handles the Ethernet communication. For the SW-only solution a software library, capable of AES-GCM, is included instead of the cryptographic hardware engine. As this paper focuses on the HW-solution, the SW-solution was only superficially considered resulting in the use of a public library [36] without any further optimization.

In order to simulate a complete Ethernet-based real-time network under all circumstances including the maximum amount of CRs without the need of connecting multiple devices, we decided to emulate the network using a second development board. From the perspective of our PLC Device Under Test (DUT), this second board can be seen as a switch sending frames from several nodes within the network. Our prototype setup is shown in Fig. 5.

VII. Evaluation of Performance Measurements

The prototype described in the previous section was used to measure performance-related timings of the cryptographic engine. We consider two different scenarios: the cryptographic engine needs to process either many minimum-size frames

or fewer maximum-size frames. The first results in a high key switching frequency, the second in a larger amount of to-be-processed IO data per communication cycle (as larger frames have a higher payload efficiency than smaller frames). A priori, it is unclear which effect has a larger impact on the performance. Moreover, both scenarios were tested for Authenticated Encryption and Authentication-Only. All considerations and measurements are based on a cycle time of 1 ms, although as stated above, the results are qualitatively independent of this concrete choice.

To ensure data comparability between the HW-only and SW-only solution, it was decided to measure times including the DMA transfer between PS and FPGA. A strong correlation between DMA throughput and payload size was identified. Due to this, the input data to the cipher core for the individual CRs was concatenated resulting in one big transfer instead of multiple small transfers. The results given in Table II only consider this concatenation approach for the inputs and not for the outputs (from FPGA to processing system). Preliminary measurements indicate a significant performance boost by concatenating the output data as well.

Results of the performance measurements are given in Table II where t_{total} denotes the measured time from the start of the DMA transfer until the last result was received, divided by the amount of CRs.

It shall be noted that this also includes some application-related time for handling the destination buffer. The DMA controller requires an uncached area for the destination buffer or an explicit cache handling. Trials showed that the cache-related operations take noticeable time, but changing to uncached area resulted in worse overall system performance. Due to this, we decided to leave the destination buffer in a cached area of the memory.

All times were measured within the processing system (cf. Fig. 4). Due to the plausibility of the measured results, we decided to not explicitly verify the time within the hardware cryptographic engine. Therefore the given times for processing the data within the hardware cryptographic engine (t_{bare}) are based on ModelSim results.

The results very clearly show that, given a suitable cryptographic hardware engine, industrial fieldbus communication in combination with AES-GCM-128 is feasible without any limitation. The results prove, both by simulation and by measurements, that the maximum allowed time for cryptographic operations per to-be-processed frame (t_{max}) was undercut even with included DMA transfer times which might be further optimized. As this is highly architecture-specific, it makes sense to focus on t_{bare} in Table II. Considering this calculated time (t_{bare}) only, it can be stated that the chosen hardware solution can outperform the required performance (t_{max}) by a factor of more than 20.

Due to the use of a non-optimized software library, the results shall only be considered qualitatively, as performance optimizations are likely to be realized. Nevertheless, it can be stated that with our setup a SW-only solution would not fulfill the requirements without limitation. To completely fulfill the requirements, a performance increase by a factor of nearly two is to be given by a dedicated CPU for this purpose. However, these results show that a SW-only solution cannot directly be excluded. For example, the authentication-only protection of 51 bytes payload with the further restriction of using only 20% of CPU time for it, would still be sufficient to run 10 CRs in parallel on our test setup.

Table II
MEASURED TIMES PER PROCESSED IO DATA FRAME GIVEN IN μs WITH THE DESCRIBED PROTOTYPE USING THE CRYPTOGRAPHIC HARDWARE ENGINE WITH A CLOCK OF 200 MHZ AS WELL AS THE SOFTWARE SOLUTION WITH A CLOCK OF 667 MHZ FOR AUTHENTICATED ENCRYPTION (AUTH. ENC.) AND AUTHENTICATION ONLY (AUTH. ONLY). t_{TOTAL} EXTENDS t_{BARE} BY THE TIME NEEDED FOR THE DMA TRANSFERS AS DESCRIBED WITHIN SECTION VII AND IS THEREFORE ONLY REQUIRED FOR THE HW-SOLUTION.

	Type	PT	AAD	$t_{\text{total}}[\mu s]$	$t_{\text{bare}}[\mu s]$	$t_{\text{max}}[\mu s]$
Auth. Enc.	HW	1444	7	13.00	5.23†	125.00
	SW	1444	7	-	243.50	125.00
	HW	44	7	2.70	0.39†	8.77
	SW	44	7	-	12.60	8.77
Auth. Only	HW	0	1451	9.30	4.27†	125.00
	SW	0	1451	-	165.30	125.00
	HW	0	51	2.60	0.36†	8.77
	SW	0	51	-	10.00	8.77

†Based on ModelSim results.

VIII. CONCLUSION AND OUTLOOK

We determined timing boundaries for cryptographic computations in PLCs that facilitate the protection of the PLC's cyclic real-time communications. We were able to show that using a simple and lightweight hardware implementation of AES-GCM, the number of simultaneously maintainable communication relations is limited by the network interface (assuming 100 Mbits/s Ethernet link) and not by the performance of the cryptographic engine. Note that this finding is independent of the used cycle time.

The situation is different if a pure software implementation of AES-GCM is used. In this case, the number of simultaneously manageable communication relations in fact *is* likely limited by the cryptographic engine. There is, however, a strong dependence on the chosen platform and a number of other implementation details. Therefore, it should not be taken as a general statement about software-only implementations.

A full implementation of the cryptographic engine in hardware might be costly and can hardly be ported to existing systems, but it clearly offers the best performance. In terms of existing systems the SW-only approach might be viable if enough computing power is available. However, general-purpose processors, especially in the embedded territory, are not typically well-suited for calculation demanded by AES. Given a modern and powerful CPU that supports an AES-specific instruction set, this approach enables a powerful approach which is interesting both for extending security features into existing systems and developing new systems respectively.

We plan to extend our study in a number of ways in the future, e.g., including other cryptographic algorithms (like message authentication algorithms based on SHA-3 [37]), evaluating the prospects of software implementations on modern industrial platforms, and investigating on suitable hardware architectures.

ACKNOWLEDGMENT

This work is based on a master thesis written at the Albert Ludwig University of Freiburg within the intelligent embedded microsystems (IEMS) program and executed at the ivESK of Offenburg University of Applied Sciences. It also has significantly been informed by our participation in the Security Working Group of PROFIBUS & PROFINET International. We would like to thank the members for fruitful discussions.

REFERENCES

[1] D. Dzung, M. Naedele, T. Von Hoff, and M. Crevatin, "Security for industrial communication systems," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1152–1177, 2005.

[2] T. Müller, A. Walz, M. Kiefer, H. Dermot Doran, and A. Sikora, "Challenges and prospects of communication security in real-time ethernet automation systems," in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, 2018, pp. 1–9.

[3] "Security Extensions for PROFINET – PI White Paper for PROFINET," https://de.profibus.com/index.php?eID=dumpFile&t=f&f=87331&token=813b3b162034298abe7b5660d6cdf22260f7a73b, Mar. 2019.

[4] F. Klasen, V. Oestreich, and M. Volz, *Industrial Communication with Fieldbus and Ethernet*. VDE-Verlag, 2011.

[5] B. Czybik, S. Hausmann, S. Heiss, and J. Jasperneite, "Performance evaluation of mac algorithms for real-time ethernet communication systems," in *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, 2013, pp. 676–681.

[6] M. Runde, C. Tebbe, and K.-H. Niemann, "Performance evaluation of an it security layer in real-time communication," in *2013 IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA)*, 2013, pp. 1–4.

[7] "IEEE standard for local and metropolitan area networks: Media access control (MAC) security," *IEEE Std 802.1AE-2006*, pp. 1–150, 2006.

[8] "Innovations in Ethernet Encryption (802.1AE -MACsec) for Securing High Speed (1-100GE) WAN Deployments," https://www.cisco.com/c/dam/en/us/td/docs/solutions/Enterprise/Security/MACsec/WP-High-Speed-WAN-Encrypt-MACsec.pdf, 2016.

[9] S. Gueron and R. Shemy, "Two are better than one: Software optimizations for AES-GCM over short messages," in *Information Technology - New Generations*, S. Latifi, Ed. Cham: Springer International Publishing, 2018, pp. 187–191.

[10] "Federal Information Processing Standards Publication 197: Announcing the Advanced Encryption Standard (AES)," https://www.cisco.com/c/dam/en/us/products/collateral/security/anyconnect-secure-mobility-client/fips.pdf, NIST, Tech. Rep., Nov. 2001.

[11] "National Institute of Standards and Technology Special Publication 800-38D: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC," https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf, NIST, Tech. Rep., Nov. 2007.

[12] Y. Sheffer, R. Holz, and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)," RFC 7525, May 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7525.txt

[13] D. H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," RFC 2104, Feb. 1997. [Online]. Available: https://rfc-editor.org/rfc/rfc2104.txt

[14] "National Institute of Standards and Technology Special Publication 800-38A: Recommendation for Block 2001 Edition Cipher Modes of Operation," https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf, NIST, Tech. Rep., Dec. 2001.

[15] "IEC 61784: Industrial communication networks - Profiles."

[16] "IEC 61158: Industrial communication networks - Fieldbus specifications."

[17] "Vergleich der Feldbussysteme," https://www.feldbusse.de/Vergleich/vergleich_feldbusse.shtml, accessed at 2021-05-04.

[18] "Industrial Ethernet – Lösungskonzeptee," https://www.feldbusse.de/Vergleich/loesungen_ethernet.shtml, accessed at 2021-05-04.

[19] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," *Journal of Cryptology*, vol. 21, pp. 469–491, 2008.

[20] P. Rogaway, "Authenticated-encryption with associated-data," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, ser. CCS '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 98–107. [Online]. Available: https://doi.org/10.1145/586110.586125

[21] D. McGrew, "An Interface and Algorithms for Authenticated Encryption," RFC 5116, Jan. 2008. [Online]. Available: https://rfc-editor.org/rfc/rfc5116.txt

[22] D. A. McGrew and J. Viega, "The security and performance of the galois/counter mode (gcm) of operation," in *Progress in Cryptology - INDOCRYPT 2004*, A. Canteaut and K. Viswanathan, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 343–355.

[23] E. Käsper and P. Schwabe, "Faster and timing-attack resistant AES-GCM," in *Cryptographic Hardware and Embedded Systems - CHES 2009*, C. Clavier and K. Gaj, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1–17.

[24] T. Krovetz and P. Rogaway, "The software performance of authenticated-encryption modes," in *Fast Software Encryption*, A. Joux, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 306–327.

[25] K. M. Abdellatif, R. Chotin-Avot, and H. Mehrez, "AES-GCM and AEGIS: Efficient and high speed hardware implementations," no. 88, pp. 1–12, 2017.

[26] T. Müller and H. D. Doran, "PROFINET real-time protection layer: Performance analysis of cryptographic and protocol processing overhead," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2018, pp. 258–265.

[27] T. Müller and H. Dermot Doran, "Protecting PROFINET cyclic real-time traffic: A performance evaluation and verification platform," in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, 2018, pp. 1–4.

[28] P. (PNO), "Profinet system description technology and application." [Online]. Available: https://profibus.com/index.php?eID=dumpFile&t=f&f=82430&token=7cbb78f5ba6b3e17762ab594f803f1901eb24fdf

[29] G. Bertoni, L. Breveglieri, P. Fragneto, M. Macchetti, and S. Marchesin, "Efficient software implementation of aes on 32-bit platforms," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2002, pp. 159–171.

[30] Y. Sovyn, V. Khoma, and M. Podpora, "Comparison of three CPU-core families for IoT applications in terms of security and performance of AES-GCM," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 339–348, 2020.

[31] S. Didla, A. Ault, and S. Bagchi, "Optimizing aes for embedded devices and wireless sensor networks." in *TridentCom*. Citeseer, 2008, p. 4.

[32] K. Akdemir, M. Dixon, W. Feghali, P. Fay, V. Gopal, J. Guilford, E. Ozturk, G. Wolrich, and R. Zohar, "Breakthrough AES performance with intel AES new instructions," *White paper, June*, p. 11, 2010.

[33] E. Homsirikamol, W. Diehl, A. Ferozpuri, F. Farahmand, P. Yalla, J.-P. Kaps, and K. Gaj, "CAESAR hardware api." *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 626, 2016.

[34] "AES-GCM high-speed implementation," Jun. 2016. [Online]. Available: https://cryptography.gmu.edu/athena/sources/2016_06_30/AES_GCM_GMU.zip

[35] XILINX, "Zynq architecture: Zynq 14.2 version," 2012. [Online]. Available: http://www.ioe.nchu.edu.tw/Pic/CourseItem/4468_20_Zynq_Architecture.pdf

[36] Markus Kosmal, "SharedAES-GCM: Attempt for a cross platform AES-GCM encryption," 2014. [Online]. Available: https://github.com/mko-x/SharedAES-GCM/

[37] "National Institute of Standards and Technology Special Publication 800-185: SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash," https://doi.org/10.6028/NIST.SP.800-185, NIST, Tech. Rep., Dec. 2016.

All links were followed on May 06, 2021.