Secure Low Latency Communication for Constrained Industrial IoT Scenarios

Jens Hiller, Martin Henze, Martin Serror, Eric Wagner, Jan Niklas Richter, Klaus Wehrle
Communication and Distributed Systems, RWTH Aachen University, Germany
Email: {hiller, henze, serror, wagner, richter, wehrle}@comsys.rwth-aachen.de

***Abstract*—The emerging Internet of Things (IoT) promises value-added services for private and business applications. However, especially the industrial IoT often faces tough communication latency boundaries, e.g., to react to production errors, realize human-robot interaction, or counter fluctuations in smart grids. Simultaneously, devices must apply security measures such as encryption and integrity protection to guard business secrets and prevent sabotage. As security processing requires significant time, the goals of secure communication and low latency contradict each other. Especially on constrained IoT devices, which are equipped with cheap, low-power processors, the overhead for security processing aggregates to a primary source of latency. We show that antedated encryption and data authentication with templates enables IoT devices to meet both, security and low latency requirements. These mechanisms offload significant security processing to a preprocessing phase and thus decrease latency during actual transmission by up to 75.9 %. Thereby they work for well-established security-proven standard ciphers.**

***Index Terms*—Security and privacy, Internet of Things, Cyber-physical Systems**

I. Introduction

In the Internet of Things (IoT), small embedded devices sense their environment to, e.g., increase the level of automation in surgeries, realize safe autonomous driving with car-to-car communication, or enable future decentralized smart grids. In the industrial domain, facilities and production lines are equipped with embedded IoT devices to increase automation, enhance monitoring, predict machine failures, track individual parts, and realize safe human-robot interaction. These advancements have the potential to significantly increase productivity and safety as well as decrease costs and production error rates [12]. To fully leverage this potential, IoT devices must exchange gathered information with other IoT devices [38], e.g., to trigger actuator actions. Especially in industrial IoT scenarios, such communication is often challenged by low latency requirements in the order of some milliseconds [23], e.g., to quickly react to production errors [2], realize safe human-robot interaction [37], or react to fluctuation in smart grids [29]. To address the challenge of low latency requirements, especially when considering the embedded and resource constrained devices in industrial IoT deployments, current state-of-the-art approaches apply special protocols with highly optimized time-slot based scheduling and priority classes [42].

While meeting latency requirements already presents a challenge on its own, the problem exacerbates considering the security requirements prevalent in the industrial IoT, e.g., to protect business secrets, prevent infiltration, and avoid sabotage [13], [15], [33]. Already providing this secure communication on its own is challenging for IoT devices due to their severely constraint resources: They face limited processing power provided by cheap and energy-efficient processors operating at a frequency of a few MHz. This limited processing power presents a challenge for secure communication as cryptographic building blocks such as encryption require significant processing times, which are already in the same order as boundaries for latency. Thus, strong communication security considerably increases communication latency. Achieving low latency but also secure communication is thus a vital but also largely contradictory goal in the industrial IoT.

In traditional Internet communication, different approaches aim to reduce the impact of secure connection establishment on latency [8], [20], [30], [31], e.g., to improve web browsing. Also in the IoT, different approaches aim to reduce high connection setup times [19], [21], which are often prohibitive for devices with low computational power. While these approaches focus on the secure connection *establishment*, we argue that the orthogonal problem of improving latency of secure communication, i.e., actual transmission of data over a secure channel, is of peculiar relevance for the resource-constrained industrial IoT: Since communication in the industrial IoT is typically periodic with a static set of communication partners, we observe a tendency for long-lived connections to overcome long connection setup times. As a result, the encryption and authentication of communicated data become major impact contributors to latency. In fact, as we show in this paper, encrypting and authenticating a message of about 650 byte using well-established and security-proven standard cryptography takes more than 20 ms on an embedded IoT device, failing to meet latency constraints of industrial scenarios such as factory maintenance, production lines, and smart grids.

In this paper, we tackle the challenge of enabling well-established secure communication in industrial low latency scenarios despite severe resource constraints of IoT devices. Our key idea is to leverage preprocessing to efficiently encrypt and authenticate payload for already established secure connections. Specifically, our contributions are as follows:

- We show that latency in secure industrial communication with long-lived connections is dominated by encryption and authentication processing (Section II) and discuss the related work on improving latency (Section III).
- We present *antedated encryption* and *fast authentication with templates*, our two approaches to decrease latency

 DOI: 10.1109/LCN.2018.8638027

for secure communication in the constrained industrial IoT (Section IV). Our evaluation shows that we are able to reduce communication latency by up to 75.97 % and enable secure low latency communication with well-established and security-proven standard ciphers (Section VI).
- Surveying latency requirements of industrial IoT use cases, we provide a classification of low latency scenarios based on communication payload sizes and corresponding latency boundaries (Section V), which is relevant also beyond the immediate scope of this paper.

II. Security Significantly Impacts Latency

The industrial IoT consists of a huge amount of IoT devices, e.g., to equip each production part with an own monitoring device. Hence, to keep deployment costs low, these devices typically base on cheap low-energy processors with limited processing capabilities, provide limited memory, employ low-power communication chips that provide 802.15.4 connectivity, and may be powered by batteries [12], [14]. To counter these constraints, these IoT devices typically operate adapted communication and security protocols that are specifically tailored to these limited resources [16], [24]. In the following, we specifically focus on the security protocols prevalent in industrial IoT deployments and show that the required processing operations for secure communication impose significant challenges when striving to stay within latency boundaries.

A. Secure Communication in the IoT

Secure communication ensures confidentiality and integrity of transmitted data using encryption and authentication. These security mechanisms can be realized at different layers of the network stack and thus different scopes of communication: Security at the transport layer establishes end-to-end security between applications, network layer security secures communication between systems or securely interconnects networks, and link layer security protects one-hop communication.

Common among security protocols across these layers is that their operation can be divided into two distinct parts: (i) secure connection setup which includes the exchange of cryptographic keys and authentication of remote peers and (ii) encryption and authentication of network packets based on the cryptographic keys derived during the connection setup. While each connection is set up only once, encryption and authentication are performed for each sent network packet.

To secure the transport layer, constrained IoT environments commonly rely on *DTLS* [32] as it can operate on unreliable transport protocols such as UDP [24] in contrast to the heavyweight reliable transport protocols used with TLS in the traditional Internet. Similarly, IoT devices that employ network layer security first set up secure connections with lightweight protocol adaptations such as *minimal IKE* or *HIP DEX*, which restrict functionality to a bare minimum to account for constraints of devices and low-power 802.15.4 networks [21], and then protect the confidentiality and authenticity of IP packets with IPsec [18]. Finally, the 802.15.4 standard [22] defines encryption and data authentication for the link layer, but relies on other means, e.g., secure out-of-bound exchange, to set up cryptographic keys and authenticate communication partners [28]. In the following, we analyze the effects of the two distinct protocol parts, i.e., secure connection setup and the protection of payload, on communication latency.

B. Connection Setup Latency

To set up secure connections, i.e., derive cryptographic keys and authenticate peers, IoT devices rely on public key cryptography. The cryptographic operations required for this pose severe challenges for constrained devices and hence lead to latency in the order of seconds [20]. This overhead renders it impossible to meet the low latency requirements of industrial IoT scenarios in the order of milliseconds (cf. Section V). As a result, IoT devices in low latency scenarios typically establish long-term connections such that the overhead of connection establishment occurs only once and before latency-sensitive data needs to be transmitted. Hence, the continuous encryption and authentication of data is the primary security-related source of latency for constrained industrial IoT devices.

C. Encryption and Data Authentication Latency

Cipher computations clearly account for the majority of processing required for encryption and data authentication. The state-of-the-art cipher, both in the Internet and the constrained IoT, is AES. AES encrypts data in *blocks* of 16 byte and employs different *block modes* to enable encryption of data of arbitrary length. Block modes use multiple AES operations and data combination strategies to transfer plaintext to ciphertext. Commonly used block modes are the *counter mode* (CTR) and the *cipher-block chaining mode* (CBC), which also features data authentication (*CBC-MAC*). Devices can combine such modes to encrypt and authenticate data. For example, CCM mode [27] combines CTR encryption with CBC-MAC for authentication and is especially suited for resource constrained IoT devices as it can be implemented extremely memory-efficient. Consequently, the security protocols in the IoT (cf. Section II-A) all support AES CCM.

Considering computation overhead, AES encrypts a 16 byte data block in multiple rounds, each requiring numerous XOR, memory shift, and memory copy operations to compute a round-based key and alter the unencrypted plaintext. Due to encryption of single blocks, an IoT device requires $\lceil len(packet)/16 \rceil$ of these heavyweight AES operations to encrypt a complete network packet. Furthermore, encryption and authentication typically require independent AES operations. Consequently, an IoT device performs two AES operations to secure each 16 byte block, one for encryption and another for data authentication. In industrial low latency scenarios, as discussed in more detail in Section V, this requires 2 - 8 AES operations for small packets (4 - 50 byte) and 32 up to 126 AES operations for larger packets (200 - 1000 byte).

A Zolertia Z1, which is a typical constrained IoT device as it is equipped with a 16MHz msp430 CPU and employs AES hardware acceleration on its CC2420 transceiver chip, requires up to 2.6 ms to protect data in small packet scenarios and even

up to 32.3 ms for scenarios with larger packets. While the Z1 provides AES hardware acceleration, many IoT devices even come without such support, further increasing the processing overhead for security operations. Thus, even after a secure connection has been established, security processing accounts for significant latency in embedded industrial IoT scenarios.

III. Related Work

We classify related work for improving security mechanisms to decrease communication latency into approaches that improve the establishment of secure connections and newly approaches based on novel fast ciphers. Furthermore, we discuss work that enables low latency medium access.

Session resumption is a mechanism to speed up secure connection establishment by re-using security information of a previous connection [9], [31]. Adoptions that address resource constraints enable the use of session resumption in the constrained IoT [21], [44]. From a different perspective, IoT devices can offload the heavyweight establishment of secure connections to a more powerful device and afterwards retrieve the security context to secure their connections [20], or distribute public key computations to multiple network entities [34], [35], [40]. Furthermore, rigorous compression of security protocol packet structures can reduce transmission times [19], especially when considering multi-hop communication in low-power local area networks. Finally, public key cryptography variants that base on elliptic curves can reduce handshake computation times [5]. All these approaches have in common that they optimize the establishment of secure connections. However, in constrained low latency IoT scenarios, connections are established only once and used over a long period of time (cf. Section II-B). Hence, in our scenario, it is more important to optimize the latency impact of encryption and data authentication, as these operations have to be performed for every single network packet.

From a different perspective, several approaches propose new ciphers that are specifically crafted for fast computation [3], [4], [17], [25], [40]. These approaches have in common that they have not gained sufficient attention with respect to cryptanalysis and attack vectors so far. However, in industrial settings, a single data breach can have disastrous effects on enterprise's reputation or cause legal liability, e.g., when attackers can obtain business secrets of clients. Furthermore, industrial setups easily exceed 10 years of operation and hence require long-term security guarantees. Thus, we argue that industrial use cases require the use of standardized, well-established, and security-proven ciphers such as AES. As a result, we deliberately focus on realizing secure low latency communication scenarios based on AES.

Apart from security processing, also the medium access layer, which handles access to the shared wireless medium, significantly impacts latency. Notably, well-established contention-based approaches, e.g., distributed coordination function in WiFi, even lead to unpredictable latencies, since the recovery from collisions highly depends on random backoff times. Orthogonal research thus aims at a low latency medium access based on scheduling approaches [6], [39], [41], [43], where the main challenge is to provide a guaranteed low latency bound of a few milliseconds. However, such approaches typically do not consider the notable time overhead required for security operations, which add an additional latency of a few milliseconds for constrained IoT devices. In this work, we focus on minimizing the impact of securing network packets on latency and thereby complement the efforts for low latency medium access.

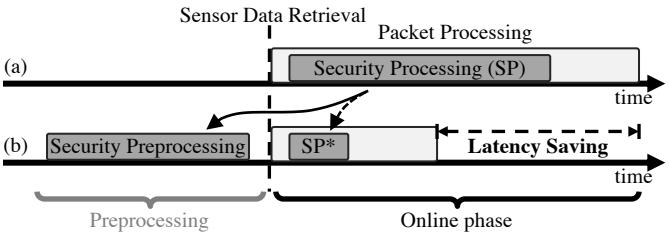

Fig. 1. Traditionally, security processing happens after sensor data retrieval (a). To save latency, we offload operations to a preprocessing phase (b) and only perform non-preprocessable operations (SP*) in the online phase.

IV. Accelerating IoT Security Processing

The processing required for securing payload significantly impacts communication latency in the industrial IoT (cf. Section II). As depicted in Figure 1a, the root cause of the resulting high latency is the security processing required for preparing data for transmission, which can only be started after the data to be transmitted, e.g., a sensor reading, is known. To realize secure low latency communication in the industrial IoT, we propose to perform significant preprocessing of security operations before data transmission is triggered (cf. Figure 1b). More specifically, we present the complementary mechanisms *antedated encryption* and *data authentication with templates*, based on the well-established and security-proven cipher AES.

Our approaches leverage the upfront knowledge of security session parameters such as cryptographic keys in long-term connections. We specifically target the most prominent block modes CTR, CBC, and CBC-MAC of AES, which also enables us to enhance CCM processing. As we detail in the following, antedated encryption pre-processes the cryptographic operations of CTR (Section IV-A), while templates speed up CBC and CBC-MAC processing (Section IV-B).

While the general possibility to preprocess CTR and CBC/CBC-MAC is well-known [7], optimizations of secure communication latency so far focused on connection establishment [5], [19]–[21], [34], [35], [40], special hardware such as GPUs to preprocess AES CTR [36], or fast instruction sets such as AES-NI [11]. In the industrial IoT with constrained resources, such hardware is unavailable and long-lived connections require the optimization of payload transfer rather than connection setup. To the best of our knowledge, the benefits of preprocessing CTR and CBC-MAC for low latency scenarios in the constrained industrial IoT have not been studied so far.

A. Antedated Encryption

With antedated encryption, we offload the heavyweight cryptographic operations of the AES CTR mode to a preprocessing phase. To this end, we leverage the property

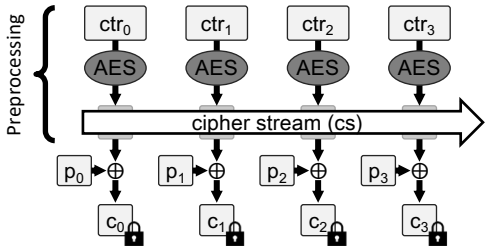

Fig. 2. Antedated encryption offloads AES operations to a pre-processing phase. Only fast XOR operations remain for the actual packet encryption.

that CTR computes a preprocessable cipher stream which is finally XORed with the plaintext to obtain the corresponding ciphertext. More specifically, a device that employs AES CTR for encryption splits the plaintext p into blocks $p_{1..n}$ of 16 byte (AES block size). To encrypt one plaintext block p_i to a ciphertext block c_i, the device uses the key k and counter values ctr_i to compute $c_i = AES(ctr_i, k) \oplus p_i$. Thereby, it sets the first counter value ctr_1 to an initialization vector (IV), which it negotiates with the communication partner, and derives further counter values ctr_{i+1} by incrementing ctr_i.

As shown in Figure 2, we split the CTR mode algorithm into two phases: (i) the heavyweight calculation of a cipherstream $cs_i = AES(ctr_i, k)$ and (ii) the lightweight final XOR step to obtain the ciphertext $c_i = cs_i \oplus p_i$. With antedated encryption, we precompute the heavyweight cipherstream creation *before* the data to be sent is known. To this end, antedated encryption leverages that the key k is set up during connection establishment and thus already known for the long-term connections. Furthermore, devices select the IV themselves or can determine it based on the security protocol specification [27]. Thus, antedated encryption can precompute the cipherstream $cs_{i=1..n}$, such that only the lightweight XOR operations are performed at the time of transmission (cf. Figure 2).

1) Security Considerations: By design, antedated encryption offers the same level of security as CTR. Most importantly, CTR requires that each pair of key and counter is only used once [27]. This property is not specific to antedated encryption and cannot be assured if the same key is used to send and receive data as counter values of messages in transit are unknown. To avoid this problem, devices must employ different keys for each sending direction, as generally applied by security protocols such as DTLS [32].

2) Selecting the Length of the Precomputed Cipherstream: For efficiently applying antedated encryption, choosing an appropriate length n of the precomputed cipherstream $cs_{1..n}$ is important, as not all protocols allow to apply a continuous cipherstream across different packets. Hence, while a too short cipherstream does not fully exhaust the potential for latency savings for longer packets, a too long cipherstream may waste computational resources. This is unproblematic for the direct application of CTR, which allows to freely choose the IV for each packet (as long as it does not repeat). In this setting, we can encrypt a packet with the first $x < n$ blocks using $cs_{1..x}$ and the next packet with the consecutive blocks starting with cs_{x+1}, i.e., using ctr_{x+1} as the IV, thus efficiently utilizing all precomputed cipherstream blocks.

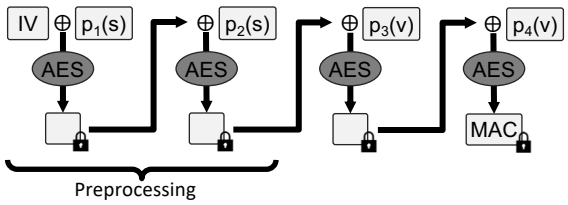

Fig. 3. We preprocess static data blocks $p_i(s)$ to require less AES operations in the online phase which then only performs AES for variable data $p_i(v)$.

In contrast, AES CCM does not allow to freely choose the IV for each packet. Instead, CCM uses a distinct part of the IV as a counter for individual blocks within one packet, which is reset for each subsequent packet, i.e., it does not count onwards across packets. Hence, a packet cannot use unused blocks of a cipherstream of a previous packet as the IV would not match. To counter this issue, we suggest to apply knowledge on past packet sizes to predict the required cipherstream size of future packets to reduce unnecessary preprocessing. As communication patterns in the industrial IoT are often repeating, past knowledge is well-suited to predict future message sizes. Alternatively, by revising the IV selection to count onwards across packets, similar to a proposal for GCM mode [36], we can enable the use of a cipherstream across different packets even when using CCM mode.

B. Fast Data Authentication with Templates

In contrast to CTR mode, CBC-MAC does not allow for a pre-computation of all AES operations, as each step of the computation requires knowledge of the transmitted data. More specifically, to obtain the CBC-MAC, the IoT device computes $c_i = AES(c_{i-1} \oplus p_i, k)$ where c_0 is set to the IV. The final ciphertext block c_n is then taken as the CBC-MAC. Consequently, this mode does not allow for precomputation as each c_i depends on the previous c_{i-1} and thus the payload p_{i-1} which contains the data that should be transmitted and is not known a priori. This is a vital property for using c_n as a MAC and, thus, cannot be relaxed to meet latency requirements: If changing any p_i would not change c_n, an attacker could modify the message in transit without invalidating the MAC.

Still, we can leverage the specific structure of messages in the industrial IoT to precompute some ciphertext blocks. More specifically, messages often consist of static data, e.g., header fields, that does not change across different messages and is thus predictable. Prominent examples include header information (we take a closer look at Goose and PPMP in our evaluation) as well as fixed parts in the encoding of sensor readings such as type-length-value encoding. As we show in Figure 3, these fixed bytes form blocks at the beginning of a message (*templates*) which we process before the actual data transmission is triggered. The length t of such a template is determined by the position of the first byte in $p_{1..n}$ that is not known upfront, i.e., typically the first value obtained from a sensor reading. As AES operates on blocks of size 16 byte, a template of length t byte enables us to precompute the CBC-MAC over the first $T = \lfloor t/16 \rfloor$ blocks. Consequently, as soon as the data to be transmitted is known, the IoT device only has to compute the CBC-MAC over the remaining $n - T$ blocks.

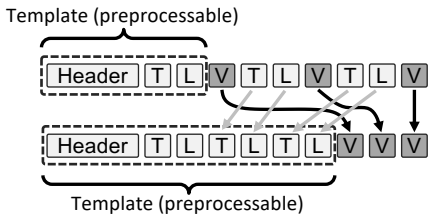

Fig. 4. Sorting fixed data to the front, e.g., in the case of type-length-value encoding, increases template size and thus also achieved latency reductions.

As IoT devices preprocess templates per packet, they can easily adapt the template to changes in the static data, e.g., protocol headers. For industrial settings, we argue that IoT devices have sufficient knowledge on packet data to derive templates for preprocessing as we will exemplify for the Goose and PPMP protocols in Section VI-B.

When applying templates to speed-up CCM, we must consider that CCM includes the message length into the IV used to calculate the CBC-MAC. As discussed in Section IV-A2, IoT devices can predict message lengths, especially when they periodically transmit the same data. Alternatively, an adapted IV construction that does not include the length would relief IoT devices from this prediction.

1) Increasing Template Size to Amplify Latency Savings: Our evaluation (cf. Section VI) shows that the template size is a critical factor that influences the efficiency of our templating approach as larger templates allow for preprocessing of a larger number of blocks and hence fraction of MAC operations. We observe, however, that the possible maximal template size is heavily impacted by the encoding of packets, e.g., if non-predictable data is interleaved with static packet parts that could – in principle – be included in a template. This occurs, e.g., when the application layer protocol encodes data in type-length-value (TLV) encoding as we illustrate in Figure 4. The standard TLV encoding (top of Figure 4) mixes static header, type, and length information with unpredictable actual data. As a result, the first value field makes it impossible to add following type and length fields to the preprocessable template. Re-sorting the packet fields (bottom of Figure 4) leads to a larger template size and potentially saves more latency. To this end, IoT devices can predict the type information from repeated reporting or request cycles and derive the length information from the typically known sensor output format (e.g., integer). The effectiveness of our proposed re-sorting strategy scales with the amount of sensor readings included in a packet, i.e., more encoded sensor readings in one packet allow us to create a larger template by re-sorting packet fields.

V. Low Latency Scenarios in the Industrial IoT

As a foundation for a detailed evaluation of antedated encryption and templating, we derive a comprehensive classification of low latency scenarios in the industrial IoT. We consider this classification to be valuable beyond our security-focused work, e.g., when designing network protocols or devising general network topologies for this domain.

We distinguish secure low latency scenarios alongside two classification metrics: (i) the *latency boundaries* that a scenario must meet and (ii) the *size of network packets* as larger packets require more security processing and thus potentially increase latency. As we assume that all scenarios use state-of-the-art cipher and key lengths, we explicitly abstract from security requirements as a potential additional classification metric.

TABLE I
Classification of IoT low latency use cases.

	small payload	**large payload**
relaxed latency	· condition monitoring [2] 50 B; 100 ms · process automation [2] 80 B; 50 - 3000 ms	· logistics automation [2] 300 B; 15 - 20 ms · factory maintenance [2] $>$ 200 B; 20 ms
low latency	· packaging [10] 15 B; 5 ms · printing [10] 30 B; 2 ms · production lines [2], [37] 20 - 50 B; 1 - 12 ms	· smart grid [29] 200 - 1521 B; 8 ms · augmented reality [2] $>$ 200 B; 10 ms

As a basis, we conducted a literature research to gather latency requirements and packet sizes of real-world scenarios. Based on the results, we identify four classes of IoT communication scenarios which we present in Table I. In the following, we discuss these four classes and the individual scenarios.

Relaxed Latency / Small Payload: First, the least challenging scenarios have comparably high latency boundaries and only need to transmit small payloads. For example, condition monitoring, a special subclass of automated diagnosis, repeatedly reports data of up to 50 byte with a latency requirement of 100 ms [2]. Similarly, process automation in chemistry or process engineering, e.g., melting or separation processes, must transmit packets of less than 80 byte within latency boundaries of 50 ms up to a few seconds [2].

Relaxed Latency / Large Payload: Such rather relaxed latency requirements initially appear less challenging for secure IoT communication. However, when constrained IoT devices must meet these latency boundaries while transmitting comparably large packets around 200 - 300 byte, security processing in the order of 10 ms already adds a large amount of latency leaving small time for other network processing and transmission. One example in this class is the automation in logistics which transmits packets of up to 300 byte under latency requirements of 15 - 20 ms [2]. Similarly, automated diagnosis and maintenance in factories transmit packets with more than 200 byte with latency boundaries starting at 20 ms [2].

Low Latency / Small Payload: The third class is characterized by low latency requirements in the order of single milliseconds. At the same time, transmitted payload is rather small, reaching up to 50 byte. These characteristics are found in production lines, printing, and packaging [2], [10], [37], e.g., for fast error reporting or coordination of machines.

Low Latency / Large Payload: Finally, the most challenging scenarios require transmission of large packets (200 byte up to a few KB) within latencies not exceeding 10 ms. A critical scenario in this class is the protection of smart grids

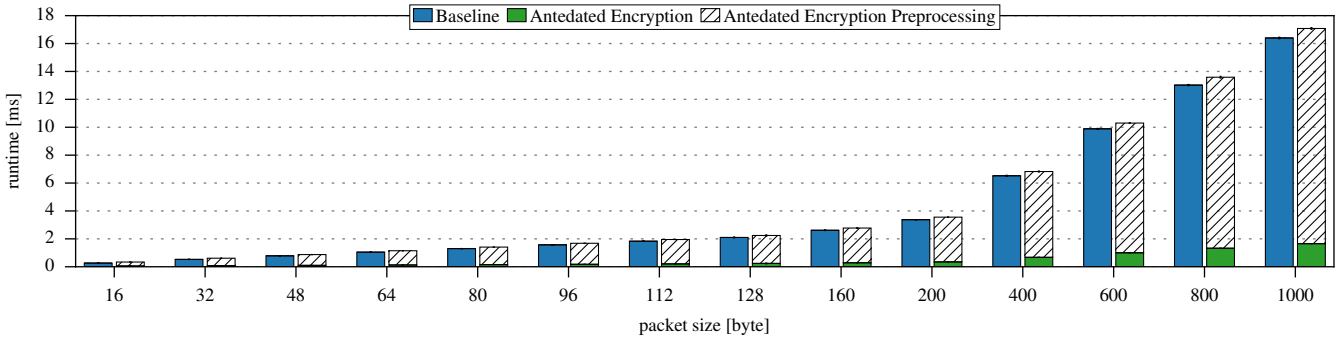

Fig. 5. Antedated encryption decreases security processing in the online phase. Latency savings increase with the packet size.

against overload [29]. Damage to parts of the grid infrastructure, e.g., caused by severe storms, can quickly propagate through the network and cause further damage, interrupting power supply and requiring time demanding costly repairs. To prevent cascading of failures and damage, grid stations notify about failures within milliseconds to timely set up protecting mechanisms. This requires packets of sizes up to 1521 byte to reach their destination within 8 ms [29]. Other scenarios in this class base upon augmented reality which only allows for communication latency of about 10 ms to keep pace with human perception [2]. Thereby, devices must transmit complex information starting at 200 byte, e.g., videos or images, to correctly augment the environment.

The various scenarios of different domains show the broad requirement for secure low-latency communication in numerous industrial applications. As we will show in our evaluation (Section VI), several scenarios of the three more challenging classes require antedated encryption and templating to meet latency requirements for given packet sizes.

VI. Evaluation

To show the effectiveness of our preprocessing approaches for low latency industrial IoT scenarios, we implemented antedated encryption and fast data authentication with templates for the AES CCM implementation of tinyDTLS and evaluate it on a Zolertia Z1 (low power msp430 CPU with 16 MHz, 8 KB RAM, 92 KB flash, CC2420 transceiver with AES acceleration). We depict the average processing time over 30 runs and show 99 % confidence intervals. As the runtime is dominated by AES operations which we offload to the hardware acceleration instead of using the software implementation of tinyDTLS, we believe that our results are transferable to other implementations. Regarding the platform, the Z1 provides a decent choice when requiring fast security processing in industrial IoT scenarios as it uses wide-spread and cheap components, hence decreasing costs for the millions of devices required for the industrial IoT. In addition, despite the cheap components, it provides AES hardware acceleration.

We first analyze the effects of preprocessing based on synthetic benchmarks considering different payload sizes and packet encodings. Thereby we separate between antedated encryption and fast data authentication with templates to highlight their respective impact on the combined latency savings. We compare each of these approaches with the respective traditional process which does not apply preprocessing (denoted as baseline). Based on the combined latency savings of both antedated encryption and templating, we finally discuss the impact on our scenarios with real-world requirements.

A. Latency Benefits From Antedated Encryption

Figure 5 shows the processing savings of antedated encryption for different packet sizes. Already for the smallest possible size of a single AES block (16 byte), antedated encryption reduces the overhead for encryption in the online phase by 78.93 % or 0.21 ms. The absolute savings increase with increasing plaintext size as more AES operations take place in the preprocessing phase. This leads to latency reductions of 1.15 ms for 80 byte payload up to 14.74 ms for 1000 byte. This property renders antedated encryption still useful for scenarios with extraordinary large packet sizes as long as latency boundaries increase in the same relative orders. The respective relative savings of 88.13 % and 89.90 % also slightly increase due to the smaller impact of CCM initialization, i.e., set up operations before processing of the actual data.

Considering energy consumption, antedated encryption essentially does not change the operations, but only relocates the time-point of their execution. Hence, energy overheads correlate with runtime overheads for storage and recovery of intermediate state. To quantify this overhead, we analyze the total runtime, i.e., combined times for preprocessing and online phase. We observe a runtime overhead of 27.59 % for a single block which rapidly decreases to 11.28 % for 48 byte and only 8.20 % for 80 byte. The overhead further decreases for increasing payload sizes, amounting to 5.36 % for 200 byte down to only 4.14 % for 1000 byte payloads. Thus, our approach incurs some energy overhead especially for IoT devices that transmit a high number of small packets. However, considering the benefits of security operation preprocessing for the challenging latency boundaries in industrial IoT scenarios, these energy overheads are reasonable.

Although the overall runtime shows an overhead, in reality, antedated encryption only decreases, but does not increase latency. We distinguish two cases: First, if packet data is not yet available, our approach preprocesses security operations and thereby reduces latency for the online phase that takes place when data becomes available. Second, if packet data becomes available before preprocessing has taken place, the

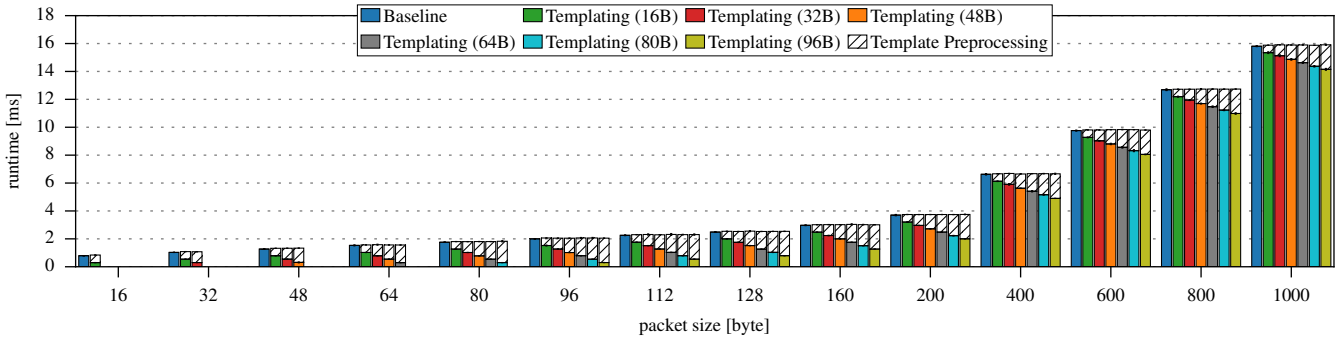

Fig. 6. Also templating decreases security processing in the online phase. The benefit increases with larger template sizes (denoted in brackets).

IoT device instead falls back to the traditional approach without a dedicated preprocessing phase and thus achieves the same latency as an unmodified system. We argue that the first case is prevalent, as IoT devices in industrial low latency scenarios know communication patterns and have sufficient spare time for preprocessing as overloaded devices would not be able to meet latency boundaries.

In summary, especially in face of the most challenging class of low latency scenarios with high packet sizes and challenging small latency boundaries, the significant latency savings of more than 89 % achieved by antedated encryption help us to realize these scenarios. We further detail the influence of these latency reductions on our use cases in Section VI-C.

B. Effectiveness of Fast Data Authentication with Templates

Templating is only interesting for larger payloads, as a benefit is only yielded if at least one 16 byte block is available for preprocessing. To get an idea for reasonable assumptions for the amount of preprocessable bytes, we analyzed protocols used in the IoT environment for larger payloads. In particular, we looked at the Goose protocol from the IEC 61850 standard [26] and the PPMP protocol [1]. While the former uses a TLV encoding, the latter uses JSON for the encoding of its messages. In both cases we were able to observe that the start of a messages is often filled with information that is already available at the time of preprocessing. These information consist of data such as a device ID, the message length, sequence numbers and a time to live indicator. In particular, we derived that anything up to 96 bytes of preprocessable bytes is reasonable, and possibly more if reordering of the packet structure is an option (cf. Section IV-B1).

To evaluate the latency reductions of templating, we thus measured MAC computation times for template sizes of 16 - 96 byte and again varied the packet size (16 - 1000 byte). Our results (cf. Figure 6) highlight the increasing latency reductions for increasing template sizes (for a fixed packet size) as a larger template enables us to move more AES operations to the preprocessing phase. For example, for a 112 byte packet a 16 byte template saves up to 0.50 ms (21.99 %) and enlarging the template size increases this saving by about 0.24 ms for each additional AES block in the template such that a 96 byte template saves 1.71 ms (75.85 %). We observe a doubled saving for the first 16 byte as we preprocess two AES blocks in the case of CCM. CCM performs an additional encryption of information such as the message length to obtain the IV for CBC-MAC initialization. We preprocess this encryption in addition to the actual template data.

Varying packet sizes do not impact the absolute latency savings, e.g., 48 byte templates always save roughly 1 ms. The relative savings, however, depend on the template's share of the full packet as this determines the relative amount of preprocessable AES operations. Thus, from a relative perspective, templating is more effective when a large proportion of the packet consists of the template, e.g., a 48 byte template saves only 5.95 % of the MAC processing time for a 1000 byte packet, but 64.45 % for a 64 byte packet.

Considering the combined runtime for preprocessing and online computation, we observe a static overhead of 0.02 up to 0.1 ms for templating. This overhead stems from additional initialization as – in the online phase – the templating approach recovers the intermediate state computed in the preprocessing phase and resumes the work. Again, as we essentially relocate the time-point of execution of operations, the energy overhead of templating correlates with this runtime overhead. Consequently, templating only slightly increases the energy consumption per packet.

Despite the total runtime overhead, templating in reality only decreases latency. To this end, similar to antedated encryption, it falls back to the traditional processing if preprocessable data is not available in time (cf. Section VI-A).

The effects of antedated encryption and templating are cumulative and, hence, their combination yields latency savings in the sum of the respective approaches. In the following, we analyze the effect of these combined latency savings on our scenarios which we discussed in Section V.

C. Preprocessing Enables Low Latency Scenarios

Finally, we tackle the question of the effectiveness of antedated encryption and templating for our low latency scenarios in the industrial IoT to show the general applicability and benefit of our approaches. To this end, we combined both approaches and measured the runtime for AES CCM and template sizes of 0 (no templating), 16, 32, and 48 byte. For each scenario, Figure 7 depicts the runtime for the standard tinyDTLS AES CCM implementation as well as the reduced runtime resulting from our optimizations. Each bar shows the runtime for the minimal packet size in this scenario (bottom) up to the runtime for the largest packet size (top). To increase

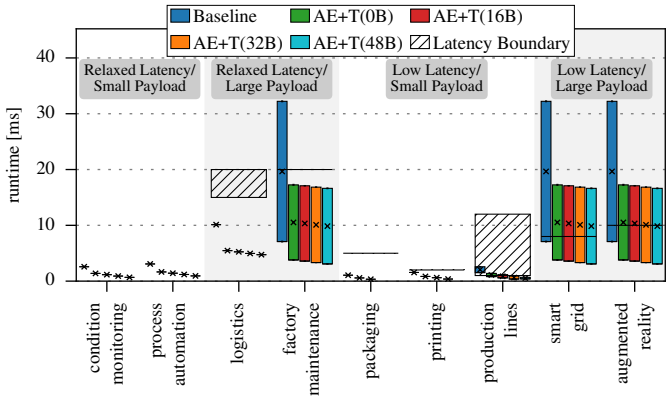

Fig. 7. Several scenarios require antedated encryption (AE) and templates (T) to meet latency requirements. For other scenarios, they increase the spare time available for non-security processing.

visibility for scenarios that observe only small differences in packet sizes which results in small bars, we mark the middle of each bar with '×'. If the template size is larger than the scenario's smallest packet size, the bottom rather shows the runtime for the template size as a template cannot be larger than the packet. In cases where the template size even exceeds the scenario's largest packet size, no bar is shown as templates of this size cannot occur in the scenario. Similarly, the hatched rectangle depicts the interval from the lowest (bottom) to the highest latency boundary (top) relevant in the respective scenario, e.g., 15 - 20 ms for logistics. In case of a single-valued latency boundary, it appears as black horizontal line, e.g., at 20 ms for factory maintenance.

The first two scenarios, i.e., condition monitoring and process automation, represent the least challenging class with high latency boundaries and small packet sizes. We do not even depict their latency boundaries, which are above 50 ms, in the figure to increase visibility. Both approaches do not require our security optimizations to fulfill their latency requirements.

The more challenging scenarios with large packets and relaxed latency, however, cannot always meet their latency requirements with out-of-the-box security approaches. For the 80 byte packets in logistics, the standard approach still can meet the latency requirements, but leaves only 4.88 ms for all other processing tasks such as packet encoding and medium access. Our antedated encryption and templating approaches increase this valuable time for further processing to 10 - 10.26 ms depending on the template size. Furthermore, our results highlight that for factory maintenance which must support 200 - 1000 byte packets, the standard approach cannot fulfill the 20 ms latency requirement for packets larger than 608 byte. However, with savings up to 56.55 % with antedated encryption and templating, IoT devices can handle all required packet sizes in this scenario within the latency boundary when employing our approaches.

Considering low latency and low packet size scenarios, already standard tinyDTLS can easily secure the 15 byte packets for the packaging scenario within the required 5 ms. However, the printing scenario becomes challenging as security processing only leaves 0.45 ms for other processing which triples when applying antedated encryption and templating to more promising 1.63 ms. Finally, standard security measures cannot realize the lower-end latency requirements of production line scenarios. With antedated encryption (and 0 byte templates), we can at least stay within latency boundaries for packets up to 32 byte. With 32 byte templates, antedated encryption and templating even fully accomplish the latency boundary for all of the scenario's packet sizes, and thus enable this former impossible use case.

The most challenging class of scenarios requires us to achieve low latency for large packets. Here, neither the standard tinyDTLS implementation, nor our optimized approaches can fulfill latency requirements for all packets. Nevertheless, antedated encryption and templating significantly increase the size of packets for which we can reach the latency goal. For the smart grid scenario, standard tinyDTLS fails to stay within latency boundaries for packets larger than 224 byte. With our optimizations, we can meet the latency requirements for packets up to 464 byte and even 496 byte for our largest measured template size. Similarly, we increase the possible size of packets for augmented reality from 288 byte possible with standard tinyDTLS up to 608 byte. These are significant steps towards realizing these scenarios.

In summary, a small number of less challenging scenarios can be realized already today with standard tinyDTLS. However, more challenging scenarios such as factory maintenance, production lines, or smart grids cannot meet their crucial latency requirements when applying standard security measures. Here, the latency reductions achieved by our approaches enable these scenarios to meet latency requirements for all packets or at least significantly increase possible packet sizes. Thereby, the scenarios are able to rely on well-known security-proven ciphers which meet the requirements of long-term security in long-living industrial deployments.

VII. Conclusion

Achieving secure low latency communication is challenging in industrial IoT scenarios that rely on cheap and thus constrained IoT devices. As we showed, already the runtime for security processing can overrun latency boundaries rendering low latency scenarios such as factory maintenance, production lines, and smart grids impossible.

We presented antedated encryption and fast data authentication with templates to offload significant security processing of well-established block ciphers such as AES to a preprocessing phase. These mechanisms achieve a reduction of security processing by up to 75.97 % for small and up to 56.55 % for larger packets. Together with our classification of low latency industrial IoT scenarios, these results show that antedated encryption and fast data authentication with templates enable several scenarios to meet their latency boundaries. For those scenarios that already meet low latency boundaries before, our approaches increase the time available at other communication layers which enables optimizations such as time-consuming compression or relaxes requirements on low latency medium access solutions.

While we focused our analysis on the send path, antedated encryption and fast data authentication with templates can also alleviate latency caused by receive path security processing. This only requires the receiver to determine initialization values for security processing which is possible when sender and receiver agree on a specific scheme to create initialization data for packet security processing. As future work, we consider it important to analyze processing cycles of industrial IoT devices to derive optimal scheduling strategies for interleaving preprocessing operations in the usual operation cycle.

In summary, the well-known concept of preprocessing has a great benefit for the new domain of industrial IoT scenarios enabling secure low latency communication with well-established and security proven ciphers.

VIII. Acknowledgements

This paper has received funding from the CONNECT project as part of the Electronic Components and Systems for European Leadership Joint Undertaking. Our work in the CONNECT project has received support from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 737434 as well as the German Federal Ministry of Education and Research (BMBF) under project funding reference no. 16ESE0154. The authors would further like to acknowledge support from the research training group "Human Centered System Security" sponsored by the German federal state of North Rhine-Westphalia as well as to thank the German Research Foundation DFG for the kind support within the Cluster of Excellence "Integrative Production Technology for High-Wage Countries". This paper reflects only the authors' views and the funding agencies are not responsible for any use that may be made of the information it contains.

References

[1] "PPMP Specification," https://www.eclipse.org/unide/specification/, last visited on July 21, 2018.
[2] I. Aktas *et al.*, "Funktechnologien für Industrie 4.0," Tech. Rep., 2017.
[3] A. Baysal and S. Şahin, "RoadRunneR: A Small and Fast Bitslice Block Cipher for Low Cost 8-Bit Processors," in *Lightweight Cryptography for Security and Privacy*. Springer, 2016, pp. 58–76.
[4] R. Beaulieu *et al.*, "The SIMON and SPECK lightweight block ciphers," in *ACM DAC*, Jun. 2015.
[5] A. Capossele *et al.*, "Security as a CoAP resource: an optimized DTLS implementation for the IoT," in *IEEE ICC*, Jun. 2015, pp. 549–554.
[6] C. Dombrowski and J. Gross, "EchoRing: A Low-Latency, Reliable Token-Passing MAC Protocol for Wireless Industrial Networks," in *IEEE EWC*, May 2015.
[7] M. Dworkin, "Recommendation for block cipher modes of operation," *NIST special publication*, vol. 800, no. 38B, p. 38B, 2005.
[8] P. Eronen *et al.*, "Transport Layer Security (TLS) Session Resumption without Server-Side State," RFC 5077, Jan. 2008.
[9] P. Eronen *et al.*, "Transport Layer Security (TLS) Session Resumption without Server-Side State," RFC 5077, Jan. 2008.
[10] A. Frotzscher *et al.*, "Requirements and current solutions of wireless communication in industrial automation," in *IEEE ICC*, Jun. 2014.
[11] S. Gueron, "Advanced Encryption Standard (AES) Instructions Set," 2008.
[12] V. C. Gungor and G. P. Hancke, "Industrial Wireless Sensor Networks: Challenges, Design Principles, and Technical Approaches," *IEEE Trans. on Industrial Electronics*, vol. 56, no. 10, pp. 4258–4265, Oct. 2009.
[13] M. Henze *et al.*, "SCSlib: Transparently Accessing Protected Sensor Data in the Cloud," in *AASNET*, Sep. 2014.
[14] M. Henze *et al.*, "Network Security and Privacy for Cyber-Physical Systems," in *Security and Privacy in Cyber-Physical Systems: Foundations, Principles and Applications*. Wiley-IEEE Press, Nov. 2017.
[15] M. Henze *et al.*, "Distributed Configuration, Authorization and Management in the Cloud-based Internet of Things," in *IEEE TrustCom*, Aug. 2017.
[16] J. Hiller, "End-to-End Security for Internet-Connected Smart Objects," *PIK*, vol. 36, no. 1, Feb. 2013.
[17] D. Hong *et al.*, "LEA: A 128-Bit Block Cipher for Fast Encryption on Common Processors," in *Information Security Applications*. Cham: Springer International Publishing, 2014, pp. 3–27.
[18] R. Housley, "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)," RFC 4309, Dec. 2005.
[19] R. Hummen *et al.*, "Slimfit – A HIP DEX Compression Layer for the IP-based Internet of Things," in *IEEE WiMob*, Oct. 2013.
[20] R. Hummen *et al.*, "Delegation-based Authentication and Authorization for the IP-based Internet of Things," in *IEEE SECON*, Jun. 2014.
[21] R. Hummen *et al.*, "Tailoring End-to-End IP Security Protocols to the Internet of Things," in *IEEE ICNP*, Oct. 2013.
[22] IEEE, "Standard for Low-Rate Wireless Networks," *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, Apr. 2016.
[23] IEEE Standards Association, "Standards for Time-Sensitive Networking for use in Industrial Automation Networks: Time-Sensitive Networking (TSN) for Industry 4.0," Tech. Rep., 2017.
[24] S. L. Keoh, S. S. Kumar, and H. Tschofenig, "Securing the Internet of Things: A Standardization Perspective," *IEEE Internet of Things Journal*, vol. 1, no. 3, pp. 265–275, 2014.
[25] M. Knežević, V. Nikov, and P. Rombouts, "Low-Latency Encryption – Is "Lightweight = Light + Wait"?" in *Cryptographic Hardware and Embedded Systems*. Springer, 2012, pp. 426–446.
[26] C. Kriger, S. Behardien, and J.-C. Retonda-Modiya, "A Detailed Analysis of the GOOSE Message Structure in an IEC 61850 Standard-Based Substation Automation System," *Int'l Journal of Computers Communications & Control*, vol. 8, no. 5, pp. 708–721, 2013.
[27] D. McGrew and D. Bailey, "AES-CCM Cipher Suites for Transport Layer Security (TLS)," RFC 6655, Jul. 2012.
[28] G. Montenegro, C. Schumacher, and N. Kushalnagar, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals," RFC 4919, Aug. 2007.
[29] P. Popovski *et al.*, "Scenarios, requirements and KPIs for 5G mobile and wireless system," Tech. Rep., 2013.
[30] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," IETF, Internet-Draft draft-ietf-tls-tls13-28 (WiP), Mar. 2018.
[31] E. Rescorla and T. Dierks, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246, Aug. 2008.
[32] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security Version 1.2," RFC 6347, Jan. 2012.
[33] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, "Security and Privacy Challenges in Industrial Internet of Things," in *ACM DAC*, Jun. 2015.
[34] Y. B. Saied and A. Olivereau, "D-HIP: A Distributed Key Exchange Scheme for HIP-based IoT," in *IEEE WoWMoM*, Jun. 2012.
[35] Y. B. Saied and A. Olivereau, "HIP Tiny Exchange (TEX): A Distributed Key Exchange Scheme for HIP-based IoT," in *ComNet*, Mar. 2012.
[36] G. Schönberger and J. Fuß, *GPU-Assisted AES Encryption Using GCM*, 2011, pp. 178–185.
[37] D. Schulze, A. Gnad, and M. Krätzig, "Anforderungsprofile im ZDKI," Tech. Rep., 2016.
[38] M. Serror *et al.*, "Towards In-Network Security for Smart Homes," in *IoT-SECFOR*, Aug. 2018.
[39] M. Serror *et al.*, "Practical Evaluation of Cooperative Communication for Ultra-Reliability and Low-Latency," in *IEEE WoWMoM*, Jun. 2018.
[40] M. B. Shemaili *et al.*, "A New Lightweight Hybrid Cryptographic Algorithm for the Internet of Things," in *ICITST*, 2012, pp. 87–92.
[41] W. Shen *et al.*, "PriorityMAC: A Priority-Enhanced MAC Protocol for Critical Traffic in Industrial Wireless Sensor and Actuator Networks," *IEEE Trans. on Indust. Inform.*, vol. 10, no. 1, pp. 824–835, Feb. 2014.
[42] P. Suriyachai, U. Roedig, and A. Scott, "A Survey of MAC Protocols for Mission-Critical Applications in Wireless Sensor Networks," *IEEE Communications Surveys Tutorials*, vol. 14, no. 2, pp. 240–264, 2012.
[43] V. N. Swamy *et al.*, "Cooperative Communication for High-Reliability Low-Latency Wireless Control," in *IEEE ICC*, Jun. 2015.
[44] T. Zimmermann *et al.*, "SPLIT: Smart Protocol Loading for the IoT," in *EWSN*, Feb. 2018.