MAC Aggregation over Lossy Channels in DTLS 1.3

Eric Wagner*†, David Heye†, Jan Bauer*, Klaus Wehrle†, Martin Serror*
*Cyber Analysis & Defense, Fraunhofer FKIE · {firstname.lastname}@fkie.fraunhofer.de
†Communication and Distributed Systems, RWTH Aachen University · {lastname}@comsys.rwth-aachen.de

***Abstract*—Aggregating Message Authentication Codes (MACs) promises to save valuable bandwidth in resource-constrained environments. The idea is simple: Instead of appending an authentication tag to each message in a communication stream, the integrity protection of multiple messages is aggregated into a single tag. Recent studies postulate, *e.g.*, based on simulations, that these benefits also spread to wireless, and thus lossy, scenarios despite each lost packet typically resulting in the loss of integrity protection information for multiple messages. In this paper, we investigate these claims in a real deployment. Therefore, we first design a MAC aggregation extension for the Datagram Transport Layer Security (DTLS) 1.3 protocol. Afterward, we extensively evaluate the performance of MAC aggregation on a complete communication protocol stack on embedded hardware. We find that MAC aggregation can indeed increase goodput by up to 50 % and save up to 17 % of energy expenditure for the transmission of short messages, even in lossy channels.**

***Index Terms*—MAC aggregation, DTLS 1.3**

I. Introduction

The (industrial) Internet of Things (IoT) increasingly relies on wireless communication to interconnect sensors and actuators in a variety of domains, such as manufacturing, smart cities, or environment monitoring. The limited capacity of the wireless transmission medium, shared among a growing number of devices, imposes stringent bandwidth constraints on these scenarios [32]. Moreover, since IoT devices are often battery-powered, minimizing communication as the primary driver of energy consumption is critical [29].

Consequently, optimizing the utilization of limited transmission resources remains an ongoing research challenge [26]. In this context, different streams of research explore techniques like compact protocol design [22], [23], data compression [21], data aggregation [20], or in-network processing [5] to reduce bandwidth consumption. Furthermore, complementary approaches such as Hybrid Automatic Repeat Request (HARQ) [1] or relaying [10] aim at enhancing transmission reliability, thereby decreasing the need for retransmissions over lossy channels to further improve bandwidth efficiency.

At the same time, wireless channels amplify the need to secure transmitted messages as all communication is broadcast over a shared medium [27]. To meet the generally recommended 128-bit security levels for integrity protection [18], each message must include a 16-byte authentication tag, known as a Message Authentication Code (MAC). This overhead poses a significant challenge in bandwidth-constrained scenarios, where typical message sizes are only a few bytes [34]. To mitigate this challenge, low-bandwidth protocols such as LoRaWAN [16] and Sigfox [6] reduce the security level by truncating authentication tags to 32 bits and 16–40 bits, respectively, well below the 64-bit minimum security level recommended by NIST already in 2016 [17].

Recent theoretical and simulative results highlight the promising potential of MAC aggregation over bandwidth-constrained wireless channels [34]. In its simplest form, MAC aggregation involves computing a joint authentication tag over n consecutive messages, which is then appended only to the n-th message [11]. Thus, the average overhead of integrity protection per message is reduced by sharing it among multiple messages. On the downside, if any of these n messages are lost, the integrity of none of the aggregated messages can be verified. Despite this limitation, substantial goodput gains have been predicted for realistic wireless channels [34].

Nevertheless, before these promising results can be applied in the real world, two fundamental challenges must be solved. First, we need to understand how MAC aggregation can be seamlessly integrated into existing communication protocols. Here, we need to deal with delayed authentication as well as the selection of aggregation parameters, *e.g.*, the number of messages whose MACs are aggregated. Secondly, since simulations, as used by previous work, hardly reflect the dynamic channels seen in, *e.g.*, manufacturing environments, it is crucial to assess the true potential of MAC aggregation in a real deployment. Therefore, holistic evaluations of a complete communication protocol stack (*e.g.*, to fully consider the overhead of each layer) under realistic channel behavior on actual hardware are required.

To the best of our knowledge, this paper is the first to investigate the applicability of MAC aggregation over lossy channels in a physical deployment. Therefore, we integrate MAC aggregation into the Datagram Transport Layer Security (DTLS) 1.3 protocol. We decided to focus on DTLS 1.3 as it is a recent general-purpose security layer optimized for bandwidth-constrained networks, with the most compact header being only 2 byte long. Thus, DTLS 1.3 will likely be used in many future communication stacks for bandwidth-constrained communication in diverse domains and applications. Nonetheless, our insights will also shed light on the expected potential of MAC aggregation into other protocols.

For our DTLS 1.3 integration, we define an extension for the DTLS 1.3 handshake to agree on using MAC aggregation. This design enables the use of MAC aggregation with various existing and future cipher suites. Moreover, we show how dynamic parameter updates allow adapting to changes in the transmission medium. Finally, we offer an optional interface to retrieve received but not fully authenticated data optimistically

to enable the vision of progressive authentication [2]. We show that MAC aggregation seamlessly integrates into DTLS 1.3 without breaking backward compatibility and that goodput can thereby be significantly improved. Simultaneously, energy and bandwidth usage shrink in real-world deployments.

Contributions. To realize MAC aggregation for real-world deployments, we make the following contributions:

▶ We design an extension to negotiate and use MAC aggregation in the DTLS 1.3 protocol.

▶ We develop a dynamic parameter selection scheme to adapt the MAC aggregation scheme to changing channels.

▶ We evaluate MAC aggregation in a real-world deployment and improve goodput by up to 50 % while cutting energy consumption by up to 17 %.

II. Bandwidth Saving Concepts

Optimizing bandwidth utilization in resource-constrained wireless networks has been tackled from various angles. In the following, we briefly overview different optimization concepts explored over the years. Afterward, we look at the different MAC aggregation schemes that have been proposed recently.

A. Alternative Approaches

Efficient communication in resource-constrained wireless networks must carefully balance the trade-offs between bandwidth utilization, computational power, and energy consumption. One approach to optimize bandwidth utilization is to reduce transmission overhead while maximizing the amount of information transmitted per packet [31]. This can be achieved, for example, through data compression and aggregation [12], as well as by using piggybacking strategies, which include acknowledgments within data packets [24].

Other approaches aim to improve transmission reliability. Hybrid Automatic Repeat Request (HARQ), for example, combines Automatic Repeat Request (ARQ) with Forward Error Correction (FEC) to enhance transmission reliability [1]. The general idea is that the receiver stores corrupted packets and combines them with retransmitted versions to maximize the number of correctly decoded packets. HARQ can thus enhance reliability at the cost of increased end-to-end delay.

In a similar sense, cooperative communication [10] leverages spatial diversity by sharing transmission resources, *e.g.*, antennas, to enhance bandwidth efficiency. This approach requires participating devices to coordinate and cooperate, *e.g.*, by relaying overheard messages to the intended receiver. Relaying can significantly improve reliability depending on its implementation and use cases [28], which, in turn, reduces the number of retransmissions and lowers energy consumption.

However, incorporating security into resource-constrained wireless networks typically increases transmission overhead [14]. A key challenge lies in mitigating the substantial overhead introduced by MACs, particularly in narrow-bandwidth IoT communication. To address this, the following section reviews related work on MAC aggregation techniques aimed at improving bandwidth efficiency.

B. MAC Aggregation Schemes

In recent years, various MAC aggregation schemes have been proposed to split tags over multiple messages [19], to progressively authenticate messages [2], [7], [15], or to minimize the effect of packet loss [33]. These schemes are described by dependency sets, *i.e.*, which messages are protected with one tag, and an aggregation function, *i.e.*, how the tags are aggregated together. We focus on two schemes, $\texttt{Agg}(\cdot)$ and $\texttt{R2D2}(\cdot)$, with XOR as their aggregation function and dependency sets that focus on different features. $\texttt{Agg}(\cdot)$ minimizes verification delays while $\texttt{R2D2}(\cdot)$ thwarts selective jamming attacks. While we use XOR as the aggregation function due to its efficiency, other provable-secure aggregation functions, such as the one used by Whips [2], can also be considered. While pseudorandom MACs, *e.g.*, HMAC, can be securely aggregated using XOR, some Carter-Wegman MAC constructions, *e.g.*, GMAC or Poly1305, are not provably secure when aggregated under XOR [4]. The schemes considered in the rest of this paper are detailed below:

$\texttt{Trad}$: To quantify the performance of existing MAC aggregation schemes, we compare them to the baseline performance of a generic traditional MAC scheme. Therefore, we consider a MAC that authenticates each message m_i with an individual tag t_i. To achieve 128-bit security, this tag is 16 byte long.

$\texttt{Agg}(n)$: The most prominent aggregation scheme is aggregated MAC $\texttt{Agg}(\cdot)$ as introduced in 2008 [11] and also later extended to prevent reordering attacks [4], to allow messages to occur multiple times [13], and to identify faulty messages in an aggregate [9]. For these schemes, an (aggregated) tag t^{agg} is appended to every n-th message, where n is the parameter for how many messages' authentication tags are aggregated. For every n-th message, a tag is then computed by XOR-ing the authentication tags of all considered messages:

$$t_i^{\text{agg}} = \bigoplus_{i-n<k\leq i} t_k \quad \text{for } i \equiv -1 \pmod{n}.$$

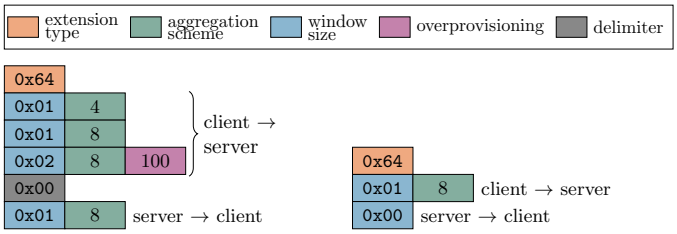
$\texttt{R2D2}(n, o)$: The idea of progressive MACs is to protect messages with initially reduced security that is improved successively as more subsequent messages are received [2], [7], [15], [25]. Randomized and Resilient Dependency Distribution (R2D2) improves upon this concept in the presence of packet loss by introducing dependency sets that bound the effect that a dropped packet can have on the verifiability of any other message [33]. If tags are 2 byte long, any other message can lose at most 16 bits of security due to a lost message. Furthermore, $\texttt{R2D2}(\cdot)$ randomizes the concrete dependency set $\mathcal{D}$ and assigns a different set to each bit of a tag to thwart selective jamming attacks. The final aggregate tag t^{agg} is thus a juxtaposition of bit-long tags and is defined as

$$t_i^{\text{agg}}[j] = \bigoplus_{0\leq k<|\mathcal{D}_j|} t_{i-\mathcal{D}_j[k]}[k \cdot |t| + j]$$

with $\mathcal{D}_j[k]$ representing the k-th entry of j-th bit's dependency set $\mathcal{D}_j$ and $t[k]$ represents the k-th bit in the tag t. To provide full security under packet loss, $\texttt{R2D2}(\cdot)$ uses overprovisioning

Identifier	Dependency Set	Aggregation Function
0x00	-	-
0x01	Agg(·)	XOR
0x02	R2D2(·)	XOR

TABLE I: The two aggregation schemes as encoded in our proposed DTLS 1.3 extension. The identifiers 0x03 – 0xff are reserved for future schemes.



(a) Extension request. (b) Extension response.

Fig. 1: During a DTLS 1.3 handshake, the client can append an extension to request MAC aggregation. The structure of this request is shown in Figure 1a and contains a list of supported aggregation schemes and parameters for both communication directions. The response, depicted in Figure 1b, selects the concrete schemes and parameters for the session. If no MAC aggregation is required, 0x00 is used, as in this response for server to client communication.

expressed as a factor o. This factor defines, in percent, how much security may be extended beyond the target, *i.e.*, $o = 100$ means that messages are protected by 256-bit security instead of the target of 128-bit security at the expense of longer tags. We fix the number of aggregated tags n to 8 as this value performs best according to prior experimentation [34].

III. Integrating MAC Aggregation into DTLS 1.3

Recent results suggest the benefits of MAC aggregation carry over to wireless scenarios, when bandwidth is constrained and messages are short [34]. To investigate this potential in a real-world setting, we set out to integrate MAC aggregation into DTLS 1.3 [23]. DTLS 1.3 specifically introduces many optimizations to save bandwidth in constrained environments and thus offers a suitable target for this investigation. In the following, we present how we integrate MAC aggregation without breaking backward compatibility.

A. Handshake Extension

The DTLS 1.3 standard defines a handshake extension to implement new functionality. We use this extension for clients to find out whether the server they connect to supports MAC aggregation and then to agree on a concrete aggregation scheme and parameters. Therefore, we propose to add a new ExtensionType value of *e.g.*, 0x64. We define the structure of the extension messages as a sequence of aggregation schemes and respective parameter sets for both communication directions. Each aggregation scheme is represented by a unique 1 B identifier as shown in Table I. Each aggregation scheme is defined by a dependency set and an aggregation function.

The structure of the extensions is displayed in Figure 1. Each scheme is identified by its identifier and followed by either one or two aggregation parameters (n, o) depending on the aggregation scheme. Supported aggregation schemes are defined for both directions individually, first from server to client and then from client to server, separated by a null byte.

The Extension is initially sent by the client to the server following the ClientHello to indicate which aggregation schemes it supports. If the server does not know the Extension, the extension message is simply ignored as by the DTLS 1.3 standard [23] and a conventional DTLS 1.3 session is established as illustrated in Figure 2a.

Otherwise, Figure 2b shows how a DTLS 1.3 session with MAC aggregation is established. Following the ServerHello, the server answers with an EncryptedExtension message that indicates support for MAC aggregation. The answered EncryptedExtension contains exactly one aggregation scheme and parameter set for each transmission direction, selected from those advertised by the client.

The MAC algorithm and the desirable security levels for integrity protection are then defined by the negotiated cipher suite. In the following, we assume that TLS_AES_128_GCM_SHA256 is the agreed-upon cipher suite. Thus, AES encryption in counter mode is used without expanding the ciphertext, and a GMAC is appended.

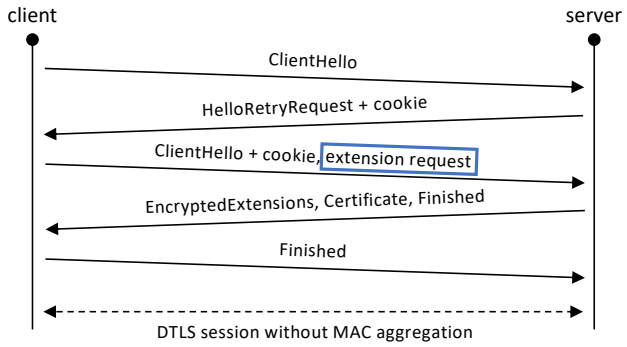
If the client and the server successfully negotiate MAC aggregation, all subsequent record layer messages, *i.e.*, content type 23, are protected with an aggregated tag. However, control messages, such as KeyUpdate messages, still contain a full MAC such that they can be immediately and fully verified and processed.

B. Record Layer Adaptations

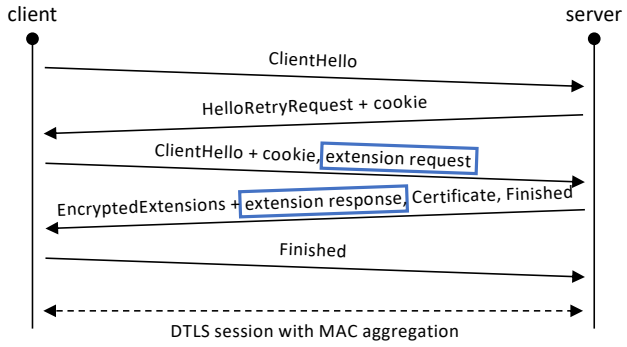
DTLS transfers data in record layer messages that contain a 16-byte authentication tag. When using MAC aggregation, authentication tags are shorter than these 16 B. Consequently, authentication tags no longer have a constant length, and ciphertexts may be theoretically as short as 2 B (an encrypted content type with 1 B payload and no authentication tag). This optimization, however, leads to two challenges regarding the identification of content types of frames and the sequence number encryption method of DTLS 1.3. In the following, we describe these two challenges as well as our proposed workarounds.

1) Content Type Extraction: Varying length authentication tags introduced by MAC aggregation make it challenging to identify non-record layer frames, such as KeyUpdates. Such packets are sent with a full 16-byte tag to be immediately verifiable with full security and minimized reaction times. However, to identify such packets, the content type must be decrypted and read.

In DTLS 1.3 ciphertexts, the content type is placed behind the payload and in front of the authentication tag.



(a) If the server does not support the MAC aggregation extension, a normal DTLS 1.3 session is established.



(b) If the server does support the MAC aggregation extension, a MAC aggregation scheme is agreed upon and used in the established DTLS 1.3 to save bandwidth, conserve energy, and increase goodput.

Fig. 2: The extension mechanism of DTLS 1.3 allows the seamless deployment of support for MAC aggregation alongside devices not supporting the extension.

As both of these fields may now vary in length, it is not easily possible to first extract the content type to know how to proceed with the message. We could resort to always guessing a record layer protocol, and only if verification fails, consider that the message may be of another type with a longer authentication tag. However, this solution gives an advantage to attackers as manipulations can no longer be clearly identified, and it increases overall processing.

Instead, we reorder the layout of DTLS 1.3 ciphertexts after MAC aggregation has been agreed upon with the second ENCRYPTEDEXTENSION message, as shown in Figure 3. Starting then, the content type of all encrypted messages is placed at the front of the ciphertext. This is possible if we limit MAC aggregation to cipher suites that do not require padding, *e.g.*, those using AES in counter mode, as the `content type` currently also serves as padding delimiter. Overall, this is only a minor limitation as the goal of MAC aggregation, *i.e.*, saving bandwidth, opposes the use of padding anyway. After our layout changes, a receiver can decrypt the first byte (or block, depending on the selected mode of operation) to learn the alleged `content type`. Based on this `content type`, the rest of the ciphertext can be processed. The entire

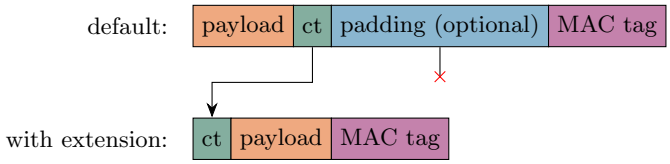


Fig. 3: After agreeing on MAC aggregation, DTLS records carry the content type as the first field to make it easily recoverable during decryption. Consequently, MAC aggregation must be used in combination with a stream cipher, *e.g.*, AES in counter mode, as the content type no longer acts as delimited between the payload and padding.

record can then be decrypted, and either the aggregated or the full authentication tag is verified.

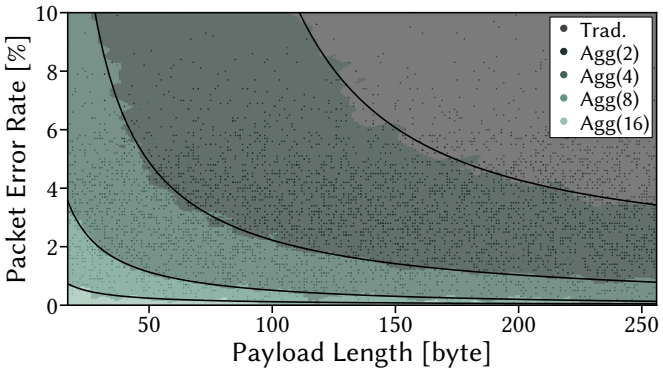
2) Sequence Number Encryption: DTLS 1.3 encrypts the sequence number in its header to protect information from third-party observers. However, because the plaintext sequence number is needed to ensure replay protection in the encryption of the payload, the sequence number is encrypted separately. Here, a mask is computed by encrypting the first 8 B or 16 B of the ciphertext (depending on whether a ChaCha or AES-based cipher suite is used) with a dedicated key. This mask is then XOR-ed with the sequence number before transmission. The sender recomputes this mask to reveal the plaintext sequence number before decrypting the message.

While the cipher suite typically ensures that the ciphertext is always longer than the required 8 B or 16 B, MAC aggregation can lead to ciphertexts that are as short as 2 B. DTLS 1.3 proposes to pad too short plaintexts for sequence number encryption. However, this would be counterproductive to the goal of MAC aggregation. Instead, we propose to pad the input to the encryption algorithm. First, we pad the input by the 8 B of the expanded epoch number such that no evident collisions occur within different epochs. If the input is still too short, the missing bytes are padded with null bytes.

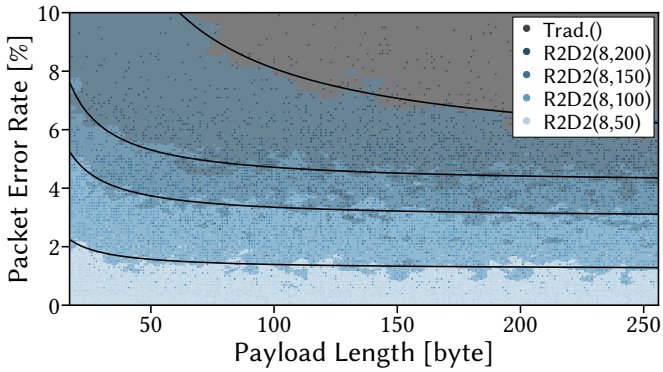
Ultimately, this input to compute the mask for sequence number encryption has a lower entropy than if a longer ciphertext is available. Hence, it may happen that the same mask is used twice during an epoch in a way that is obvious to outsiders. Even in the worst case, with 1-byte payloads and `Agg(16)` as aggregation scheme, the likelihood of a first collision rises above 50 % only after 274 messages. An observer listing in then still only learns the XOR-ed sequence number of both frames that used the same mask. Only multiple such collisions allow an attacker to infer the sequence number. However, an attacker who observes the channel for a prolonged period of time could also simply count the number of packets to gather the same information. Hence, the real-world impact of reduced entropy for sequence number encryption is low, even in extreme scenarios with minimal payloads.

C. Upper Layer Interface

In DTLS 1.3, received payload data is passed to the upper layer after its integrity has been verified. Duplicate receptions



(a) Aggressive MAC aggregation with `Agg(·)` is only optimal for short payloads and with low PERs.



(b) `R2D2(·)` can aggregate more aggressively for longer payloads but it is rarely recommendable for PERs above 8 %.

Fig. 4: We approximate the optimal aggregation parameters, *i.e.*, those yielding the highest goodput, based on boundaries described by formulas of the form $y = a \cdot e^{\frac{b}{x}} + c$, where x is the payload lengths, y is the current Packet Error Rate (PER), and a, e, c as well as b are parameters.

are discarded. Invalid messages (*e.g.*, due to an invalid authentication tag) are either silently discarded or responded to with a fatal alert. For most situations, silently discarding invalid messages is recommended to minimize the need for expensive handshakes. With MAC aggregation, most application data records cannot be fully verified upon reception. Here, the receiving entity can decide how messages should be handed to the upper layer, either by buffering data until fully verified or by supplying data optimistically.

1) Buffering Data Until Full Verification: To match the behavior of conventional DTLS 1.3, unverified data could be buffered by the DTLS layer until its integrity is fully verified. Application data is fully verified if a security level equivalent or higher to the baseline of the agreed-upon cipher suite is reached, *e.g.*, 128-bit security.

In this mode, all application data would be buffered until verified. If a tag verification fails, the receiver can either silently discard the corresponding data or answer with a fatal error, depending on their configuration. Moreover, buffered data is eventually discarded if data missing for verification is considered definitively lost.

2) Optimistic Data Processing: The idea of progressive message authentication is to optimistically process data before full integrity verification takes place [2]. The premise is that attacks are rare, and the benefits of low-latency data processing outweigh the cost of recovering from manipulated data that has evaded the lower security threshold. Many scenarios, such as Industrial Control Systems (ICSs), are envisioned to benefit from such optimistic security [3], [30]. With MAC aggregation, such an optimistic operation can be supported.

Therefore, the receiver needs to set a security level threshold that determines when data is passed to the upper layer, and a callback function that is invoked if data integrity for already forwarded data is refuted retroactively. This security threshold could *e.g.*, be set to 0, which would mean that all data is immediately passed to the upper layer unless tag verification explicitly fails. Keep in mind that not all application data records carry a tag, or the carried tag cannot be verified due to lost preceding packets. This threshold could also be set more conservatively to *e.g.*, 64-bit security, which corresponds to half the security provided by a full 16-byte tag and equals a chance of 1 in 2^{64} that a manipulated message gets infiltrated.

The callback function is called if authenticity is later refuted through the failed verification of a tag, and it provides the application layer with the number of processed bytes since the first malicious byte. How to process such data and if this data invalidation should lead to a fatal error to close the communication channel is scenario-dependent.

D. Dynamic Aggregation Parameters

While the aggregation scheme is fixed for a session, one may want to dynamically adjust aggregation parameters if channel quality changes. In extreme cases, a blocked line of sight could cause a high-reliability channel that usually benefits from aggressive aggregation to suddenly operate better without aggregation. Hence, we devise a mechanism to update aggregation parameters dynamically. This procedure is inspired by the DTLS 1.3 KEYUPDATE mechanism but is always triggered by the receiver, who can judge the quality of the channel more accurately.

1) Updating Aggregation Parameters: To this end, we define a new (post-)handshake AGGREGATIONUPDATE message, which either the client or the server may send during the session to request changing the window size and potentially other aggregation parameters at the peer. Similar to other handshake messages without a built-in response, such as KEYUPDATE, the AGGREGATIONUPDATE is acknowledged by the peer upon reception, and it is retransmitted by the sender if the acknowledgment (ACK) is not received before a timeout. The AGGREGATIONUPDATE is defined by a new message type and new parameters for the used aggregation scheme in the same format as in Figure 1.

Upon sending the AGGREGATIONUPDATE, the sender initializes a new decryption epoch with new decryption keys

and the desired aggregation parameters. Decryption keys are derived for a new epoch as if a KEYUPDATE were received. Since the AGGREGATIONUPDATE is a handshake message, it contains a full authentication tag, and the receiver can verify the tag and process the message immediately upon reception. All subsequent messages sent by the receiver must then use the new epoch for encryption. The decryption epoch at the receiver and the encryption epoch at the sender remain unchanged.

2) Acknowledging AGGREGATIONUPDATE*s:* The last messages sent before receiving an AGGREGATIONUPDATE may not be fully authenticated. Each aggregation scheme should, therefore, define how to authenticate the last messages from the prior epoch after an AGGREGATIONUPDATE. One challenge is, however, that a receiver may not know how many messages were sent in the prior epoch. Hence, to make the transition smoother, ACKs to AGGREGATIONUPDATEs contain the sequence number of the last datagram from the previous epoch as an additional field. If this ACK is lost, the receiver assumes that it received the last frame from the previous epoch. If this were not the case, some tag verifications during the transition period may fail. In that case, the unauthenticated data should be silently discarded.

For `Agg`$(\cdot)$, the transition is taken care of by an additional tag carrying the aggregated authentication tag from all yet unauthenticated messages with the ACK to the AGGREGATIONUPDATE. If the scheme switches to traditional authentication with one full tag per message, the first frame of an epoch carries two tags, one for the current datagram and one for all not yet authenticated prior frames.

For `R2D2`$(\cdot)$, all best-performing schemes depend on 8 prior messages and only differ by the overprovisioning factor. To finalize an epoch to switch this parameter, we send a full authentication tag over the last 17 messages (half of the possible total verification delay) with the ACK to the AGGREGATIONUPDATE. Then, the next epoch, using full MACs or `R2D2`$(\cdot)$ with adapted overprovisioning, starts.

3) Parameter Selection: Each receiver can decide when to switch parameters for the aggregation scheme based on which information it has, *e.g.*, to preempt a worsening or improving channel due to the receiver's movement. In the following, we propose a generic reactive scheme to adapt to relatively long-term changes of the wireless link.

We want to understand how the schemes perform under different channel conditions. Therefore, we generated binary packet loss traces for 10 000 channels based on the Gilbert-Elliot model based on real-world measurements [8]. For each of 10 000 smapled traces with varying packet lengths, we check which parameters lead to the best goodput. From these measurements, we observe that the best parameters mainly depend on the current payload length and packet error rate. The respective best schemes are visualized as dots in Figure 4.

Then, we train a k-nearest neighbor (KNN) classifier and use it to fill the area. However, we want avoid porting a KNN classifier to often heavly resource-constrained devices that MAC aggregation aims to relieve. Instead, we propose the following approximation of the KNN classifier.

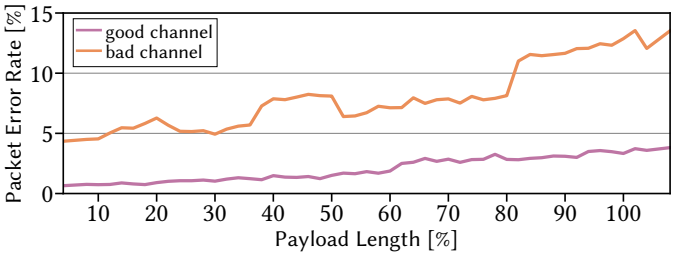


Fig. 5: As expected, the PERs increase for longer payloads as the probability of uncorrectable bit flips increases.

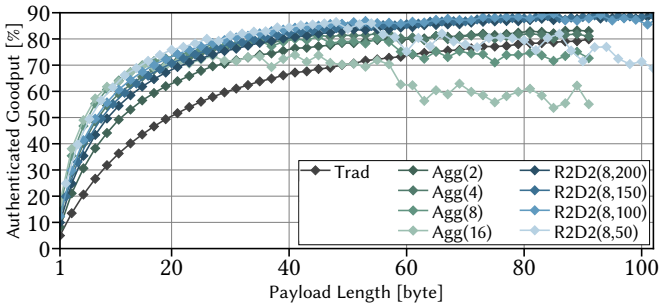
The boundaries closely resemble the behavior of a function of the form $y = a \cdot e^{\frac{b}{x}} + c$, where y is the PER and x is the payload length. Hence, we performe a least-square fitting of the boundaries between parameters to obtain the parameters a, b, and c. The resulting functions are plotted as black lines in Figure 4. The receiver can then easily compute between which boundaries the current channel performs and adapt the parameterization accordingly. To avoid unnecessary repeated switching, we propose the following mechanism to decide when to switch parameters. Based on the running average PER over the previous 200 packets, the receivers constantly assess if the currently selected parameters are optimal based on the curves identified in Figure 4. If non-optimal parameters are selected for 50 consecutive *transmitted* packets (the receiver counts lost packets once it identifies them through follow-up sequence numbers), the channel switches to the currently optimal parameters. Thus, we efficiently determine and use optimal aggregation parameters for the current channel without burdening the channel with many parameter changes.

IV. PERFORMANCE EVALUATION

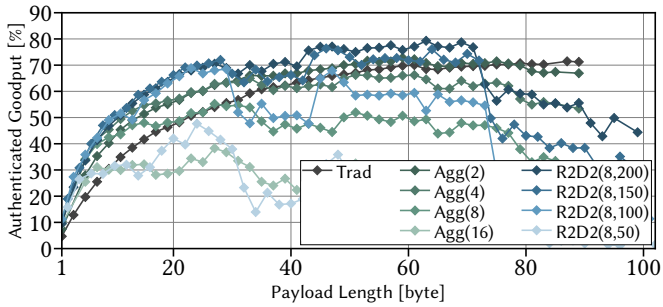
To assess MAC aggregation in realistic DTLS 1.3 scenarios, we perform a variety of tests. First, we evaluate bandwidth and energy saving for different payload lengths. Then, we dive deeper into the performance of MAC aggregation based on concrete scenarios, assessing goodput improvements, authentication delays, and dynamic aggregation parameter selection.

A. Measurement Setup

We conduct our measurements on two Zolertia RE-Mote boards equipped with a Cortex M3 @ 32 MHz, 32-bit CPU. We use the Contiki-NG operating system and adapt the wolfSSL DTLS 1.3 implementation to support MAC aggregation. The measurements are conducted in an over 1000 m^2 test and experimentation lab for energy-related equipment and components. For most measurements, the client and servers are mounted on a fence at a height of 1.94 m and 12.50 m apart. Different channel quality is achieved by changing the orientation of the antenna. Interference was caused mainly by several 802.11 networks deployed in the facility. When a stable channel was desired for comparability, measurements were conducted during the night or over the weekend, when network activity was severely reduced.

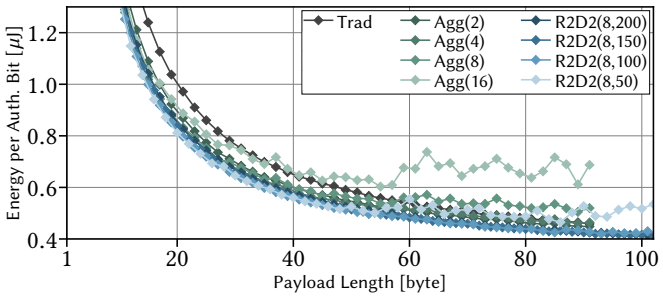


(a) With low PERs on the good channel, most aggregation schemes achieve significant goodput improvements of, *e.g.*, 32 % for 30 byte payloads.

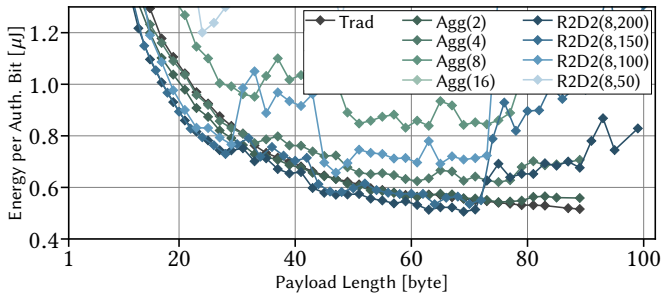


(b) Even with higher PER on the bad channel, an appropriate MAC aggregation scheme yields goodput improvements of up to 146 % for 1 byte payloads and *e.g.*, 38 % for 20 byte payloads.

Fig. 6: MAC aggregation can significantly improve authenticated goodput but can decline if applied too aggressively, especially for channels with higher PERs. On the bad channel, for example, even conservative MAC aggregation falls short of appending one long MAC to each message for payloads longer than 80 byte.



(a) MAC aggregation on the good channel with a low PER even yields 8.4 % energy saving for the longest supported payload.



(b) For short payloads, more conservative MAC aggregation can save up to 20.6 % of energy even with high PER on the bad channel.

Fig. 7: In many scenarios, MAC aggregation can lead to more than 10 % energy savings. However, the scheme and parameters must be properly chosen, especially on channels with higher PERs.

B. Aggregated MACs vs. Longer Packets

First, we assess MAC aggregation in DTLS 1.3 for a broad range of scenarios. Therefore, we collect network traces for IEEE 802.15.4 packets with payload lengths between 4 B and 108 B over a good and a bad channel. We plot the resulting PER in Figure 5. As expected, the PER increases for larger payloads. It varies between 0.65 % and 3.81 % on the good channel, and between 4.35 % and 13.52 % on the bad channel. We stitch these traces together according to the DTLS 1.3 packet sizes required by different MAC aggregation schemes for fixed payload sizes. This method approximates the performance of MAC aggregation for a wider range of scenarios than possible if measuring the scenarios individually.

1) Goodput Improvements: We first investigate the potential *authenticated goodput* improvements achieved by MAC aggregation. We define authenticated goodput as the ratio between received fully authenticated payload bytes and the overall transmitted bytes. We compare the authenticated goodput by the different schemes and traditional MACs (without aggregation) for different payload lengths in Figure 6.

On the good channel in Figure 6a, we see that MAC aggregation significantly improves authenticated goodput. The most significant gains are observed for short payloads, which is expected as the overhead of MACs is proportionally larger. We also see that the two schemes `Agg(·)` and `R2D2(·)` perform similarly with different parameters. While the most aggressive aggregations, `Agg(16)` and `R2D2(8,50)`, perform best for short messages, they also decline quickly with larger payloads due to increased packet loss (*cf.* Fig. 5). Slightly more conservative aggregation outperforms traditional MACs over the full range of 802.15.4 payload sizes. Moreover, `R2D2(·)` reduces the maximum occupied space by authentication data in a DTLS 1.3 frame, thus enabling support for longer payloads sent within individual packets (*i.e.*, 91 B vs. up to 104 B).

On the bad channel in Figure 6b, we observe that aggressive aggregations, *e.g.*, `Agg(16)` and `R2D2(8,50)`, is not effective. Still, more conservative aggregation, especially with `R2D2(·)`'s resilience to packet loss, outperforms traditional MACs up until 72-byte payloads. This falloff coincides with the channels PER raising over 8 %, above which `R2D2(·)` rarely outperforms traditional MACs as noted in Section III-D3. Overall, we still see, for example, a 38 % goodput improvement by `R2D2(8,200)` for 20-byte payloads.

We can thus conclude that MAC aggregation does indeed improve authenticated goodput. However, if the aggregation is too aggressive for a given channel, goodput quickly decreases.

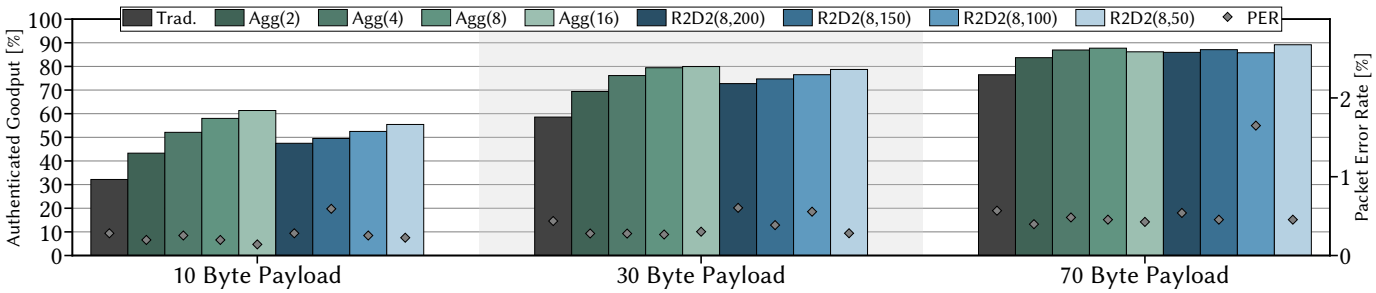


Fig. 8: Evaluating concrete scenarios, we again see that short payloads benefit most from MAC aggregation with authenticated goodput improvements of up to 90.6 %, 36.5 %, and 16.6 % for 10-byte, 30-byte, and 70-byte payloads, respectively.

Payload	Frequency	Median PER
10 B	1 Hz	0.25 %
30 B	10 Hz	0.30 %
70 B	1 Hz	0.45 %

TABLE II: Configurations of the three static scenarios.

2) Energy Consumption: In resource-constrained scenarios, the longevity of battery-operated devices may also be a critical concern. To derive the energy cost in Figure 7, we first measure the energy cost of transmitting packets of varying sizes, where we observe an expected linear increase from around 0.6 µJ for 4-byte packets to around 2.6 µJ for 110-byte packets.

Figure 7a shows that MAC aggregation saves energy for all payload lengths on the good channel. The best improvements occur for small payloads, with *e.g.*, energy saving of 27.1 % for 10-byte payloads by `R2D2(8,50)`. But also the energy cost for longer payloads is reduced. For example, `R2D2(8,150)` reduces costs by 8.4 % from 0.46 µJ to 0.42 µJ for the maximum supported 91-byte payload. These savings can be extended to 10.1 % per authenticated bit when considering that `R2D2(8,150)` supports up to 102-byte payloads.

On the bad channel in Figure 7b, we see that energy saving closely correlates with goodput improvement. Here, we see energy savings of up to 20.6 % for 21-byte payloads with `R2D2(8,150)`. However, for longer payloads above 70 B, even `Agg(2)` is outperformed by traditional MACs.

MAC aggregation leads to energy saving unless payloads are long and PERs are high. Therefore, the aggregation scheme and its parameters must, however, be appropriately chosen.

C. Static Aggregation Parameters

Now, we examine the performance of MAC aggregation based on three concrete scenarios. We analyze the achieved goodput and compare it to the results of Figure 6a before assessing the authentication delays of the different schemes.

1) Setup: To cover a variety of potential scenarios, we send 10-byte and 70-byte payloads with a frequency of 1 Hz (*i.e.*, one packet per second) and 30-byte packets with a frequency of 10 Hz, as summarized in Table II. For each of the communication scenarios, we compare the performance of traditional 16-byte MACs to the same MAC aggregation schemes as before. For each measurement of the nine schemes,

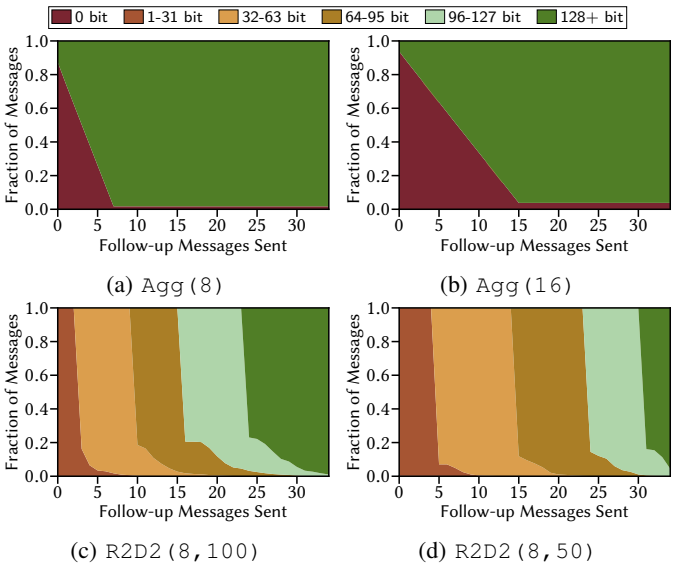
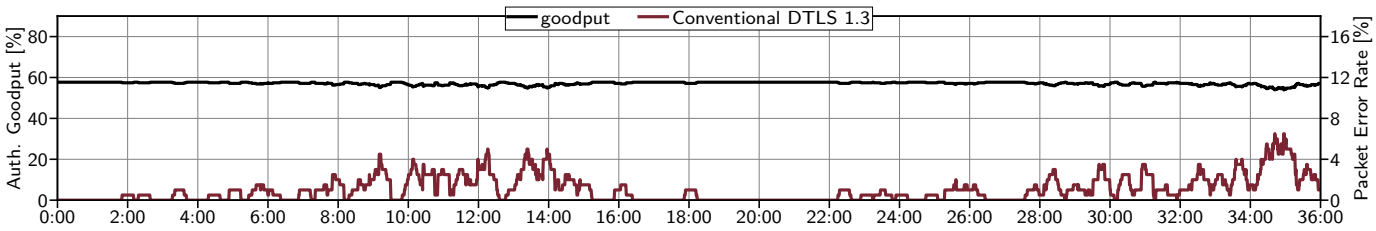


Fig. 9: `Agg(·)` provides quicker full authenticity but some data never receives any security guarantees. Meanwhile, `R2D2(·)` provides progressive authenticity improvements.

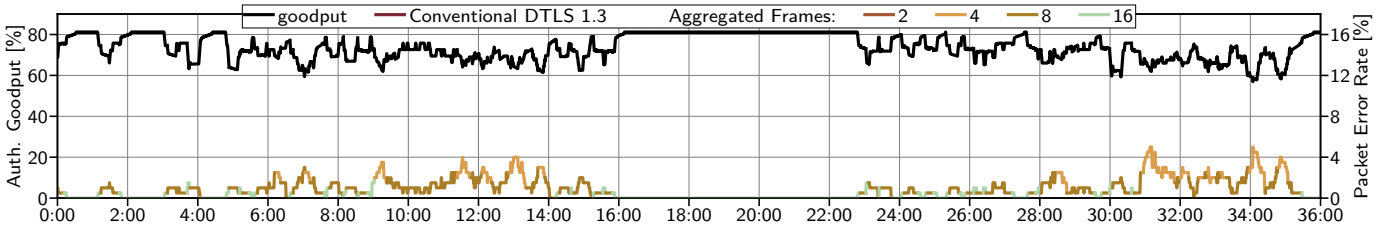
our evaluation ran for 1 hour, *i.e.*, a total of 27 hours, over a quiet weekend. The grey diamonds on the second y-axis in Figure 8 show the minor variations in PERs that can be explained by varying packet sizes, *e.g.*, scenarios with traditional 16-byte MACs yield slightly higher PERs due to the longer header. The outliers, *e.g.*, `R2D2(8,50)` for 70-byte payloads, are caused by a short burst of high packet loss.

2) Goodput: We show the achieved goodput in Figure 8. These results validate the methodology from Section IV-B and Figure 6. Thus, MAC aggregation can indeed improve goodput significantly on realistic wireless channels. For example, the authenticated goodput for 10-byte payloads achieved by traditional MAC nearly doubles from 32.17 % to 61.34 % with `Agg(16)`. These results confirm the potential of MAC aggregation for concrete scenarios and confirm theoretical and simulative results from the past in a real deployment.

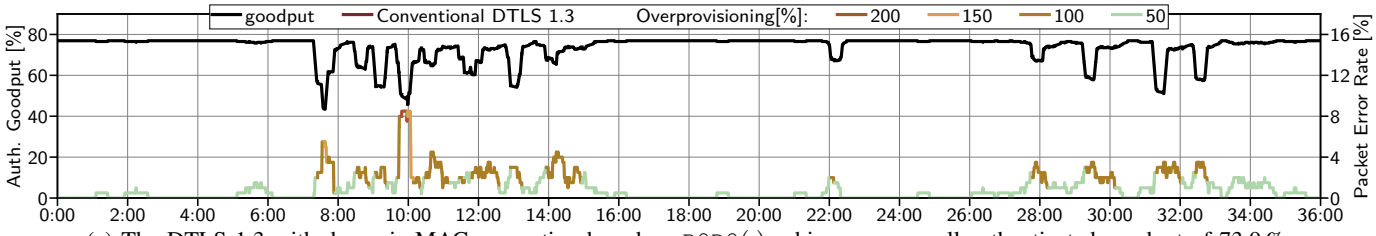
3) Authentication Delays: A valid concern about MAC aggregation schemes is authentication delay. After a packet is received, the authenticity of the payload cannot always be immediately verified. As discussed in Section III-C, our



(a) The conventional DTLS 1.3 connection achieves an overall authenticated goodput of 57.0 %.



(b) The DTLS 1.3 with dynamic MAC aggregtion based on Agg(·) achieves an overall authenticated goodput of 74.4 %.



(c) The DTLS 1.3 with dynamic MAC aggregtion based on R2D2(·) achieves an overall authenticated goodput of 73.9 %.

Fig. 10: Dynamic MAC aggregation can quickly adapt to varying channels and thus yield significant authenticated goodput improvements for short DTLS 1.3 transmission even under changing channel conditions.

integration of MAC aggregation into DTLS 1.3 offers two ways to deal with this delay. Either any not yet authenticated data can be buffered by the DTLS layer and forwarded to the higher communication layer once authenticity is guaranteed. Alternatively, progressive authentication [2] can be employed to optimistically push data to the higher layer and raise an alert in the unlikely case that a falsified MAC is detected retroactively. For both approaches, it is important to understand *how* MAC aggregation delays authenticity validation.

Therefore, we visualize this delay in Figure 9 for the 30-byte payload measurements for four MAC aggregation schemes, namely `Agg(8)`, `Agg(16)`, `R2D2(8,100)`, and `R2D2(8,50)`. The y-axis shows the fraction of messages that have been authenticated to the given security level at a specific time. The time is expressed on the x-axis in terms of sent follow-up messages after the initial message. Thus, after five follow-up messages have been transmitted with `Agg(8)`, 73.9 % of messages have achieved a satisfactory security level of 128 bit or higher (●) and 26.1 % of messages completely lack any authenticity (●), as can be seen in Figure 9a. We see that Agg(·) has lower authentication delays than R2D2(·) and that either a message is fully authenticated or not at all.

R2D2(·) behaves differently as data is progressively authenticated. Thus, upon reception, all messages are already authenticated to a minimal level. However, the same mechanisms that enable this progressiveness also lead to a longer time to reach full authenticity. Thus, R2D2(·) is advantageous in scenarios where authentication delay is less important or if data can be processed with minimal security guarantees, while quick retrospective detection of manipulations is expected.

Overall, MAC aggregation leads to authentication delays that can be managed in different ways. These delays differ vastly across aggregation schemes, so an educated choice is advised for a given deployment.

D. Dynamic Parameter Selection

In Section III-D, we presented a mechanism to dynamically adapt the parameters of the MAC aggregation scheme to current channel conditions. Now, we investigate how conventional DTLS 1.3 connections compare to dynamic MAC aggregation with Agg(·) and R2D2(·). To gather comparable results, we need a way to repeatedly alter the wireless channel conditions.

In our experiment, we want to change channel conditions based on movement as it occurs from mobile robots in industrial scenarios. To mimic this behavior, we mount the sending Zolertial RE-Mote on a *Dreame D9 Max* vacuum robot. This robot is programmed to slowly move from a distance of 1.9 m to a distance of 13.2 m in a zigzag pattern over the course of 15 min. Afterwards, the robot drives straight back to the starting point in 1 min. For each MAC scheme, the robot drives this pattern twice with a 5 min pause in between, for a total evaluation duration of 36 min. We send 10 packets per second, each carrying one DTLS 1.3 frame with 30 B of payload.

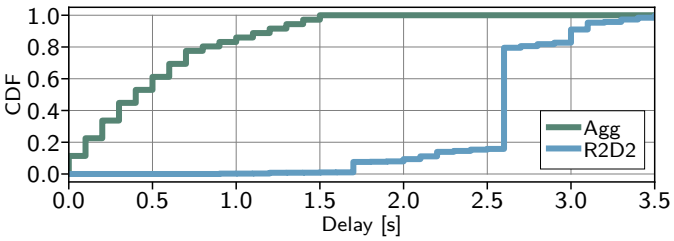


Fig. 11: Agg(·) leads to fast verification but a higher variance in delays than R2D2(·).

The results of our evaluation are shown in Figure 10. In Figure 10a, we see the behavior of a conventional DTLS 1.3 connection. The black line shows the running average of the authenticated goodput over the last 200 transmissions. Likewise, the dark red line on the secondary y-axis shows the running average PER. Overall, the PER remains low unless the robot is further away from the starting point between 7 min and 15 min, and between 28 min and 36 min. However, over time, the achieved authenticated goodput remains relatively constant slightly below 60 %. In total, the conventional DTLS 1.3 connection achieves a goodput of 57.0 %.

Figure 10b now shows the performance of Agg(·) with dynamically adjusted parameters. Here, the bottom line does not only show the running average PER on the second y-axis, but also the current MAC aggregation parameters through its color. We see that the parameters are regularly updated while the robot is in motion, selecting more conservative parameters as the channel worsens. Thereby, the authenticated goodput surpasses 80 % on a good channel. Meanwhile, the conventional DTLS 1.3 connection is also outperformed if the channel worsens, for an overall goodput of 74.4 %.

Finally, we show the results for R2D2(·) in Figure 10c. In contrast to Agg(·), R2D2(·) can keep a high authenticated goodput of close to 80 % even when PERs reach 2 %. Conversely, R2D2(·) is more severely impacted when PERs spike further. During the spike after 10 minutes, we even see that channel switching to conventional DTLS 1.3 authentication for a short while. Despite these differences to Agg(·), R2D2(·) achieves a very similar overall goodput of 73.9 %.

Figure 11 displays a CDF of the delays until data is fully authenticated for Agg(·) and R2D2(·) in this scenario. We again observe faster authentication for Agg(·), but more constant delays of mostly 2.6 s for R2D2(·).

Overall, MAC aggregation can significantly boost authenticated goodput, even in dynamic scenarios. The raw performance of the different aggregation schemes does not vary significantly, such that secondary factors, like the shorter authentication delays of Agg(·) or the support for longer payloads and resilience to selective jamming attacks by R2D2(·), should determine the choice of the MAC aggregation scheme.

V. Limitations and Future Research Potential

Our evaluations show the potential of MAC aggregation in real-world deployments. However, our evaluations also have limitations, and further optimizations are also possible. For one, potential scenarios quickly increase in complexity with multiple devices, multihop communication, and variable-size traffic in a single DTLS session. Our current evaluation setup does not cover these factors. Thus, additional research is needed to understand the full potential of MAC aggregation, as all of these factors may influence performance and thus ultimately influence whether MAC aggregation adds value to a given scenario or not.

Additionally, MAC aggregation in DTLS 1.3 and other protocols may be further optimizable in practice. The proposed integration process could be simplified or extended to offer more flexible configurations. Additionally, the potential symbiosis of reliability-increasing strategies, *e.g.*, selective retransmissions or cooperative communication, and MAC aggregation is worth exploring, as MAC aggregation greatly benefits from low PERs. Finally, the dynamic parameter selection procedure also offers potential to anticipate predictable changes to the channel quality (*e.g.*, because the distances between antennas change) or to reflect the dynamics of wireless channels better.

Exploring these future research directions thus potentially unlocks even better performance. In that sense, the results presented in this paper can be understood as a lower bound of the potential of MAC aggregation.

VI. Conclusion

MAC aggregation distributes the overhead of authentication tags over multiple messages to save valuable bandwidth, especially if transmitted payloads are short. Recent results suggest that MAC aggregation can even improve authenticated goodput on wireless and, thus, lossy channels. To the best of our knowledge, this paper presents the first real-world deployment of MAC aggregation for wireless communication.

Concretely, we show that MAC aggregation can be integrated into the DTLS 1.3 protocol while remaining standard-compliant and backward-compatible. These efforts highlight the main challenges of deploying MAC aggregation in the real world, namely the definition of the interface to the upper communication layers and the process to (dynamically) negotiate MAC aggregation schemes and associated parameters.

Finally, we extensively evaluate MAC aggregation in DTLS 1.3 for an IEEE 802.15.4 wireless channel between two Zolertia RE-Mote boards running Contiki-NG. We show that MAC aggregation indeed improves authenticated goodput significantly in many scenarios, more than doubling it for short payloads of only a few bytes. Meanwhile, this process also leads to significant energy savings and can dynamically and efficiently adapt to changing channel conditions, even falling back to full-sized MACs if PERs increases excessively. Overall, MAC aggregation can thus result in a much more efficient use of wireless communication channels.

Acknowledgments

This work was partially funded by the Federal Ministry of Research, Technology and Space (BMFTR) in Germany under the grant number 16KIS2251. The responsibility for the content of this publication lies with the authors.

References

[1] A. Ahmed, A. Al-Dweik, Y. Iraqi, H. Mukhtar, M. Naeem, and E. Hossain, "Hybrid Automatic Repeat Request (HARQ) in Wireless Communications Systems and Standards: A Contemporary Survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2711–2752, 2021.

[2] F. Armknecht, P. Walther, G. Tsudik, M. Beck, and T. Strufe, "ProMACs: Progressive and Resynchronizing MACs for Continuous Efficient Authentication of Message Streams," in *Proceedings of the Conference on Computer and Communications Security (CCS)*. ACM, 2020.

[3] J. H. Castellanos, D. Antonioli, N. O. Tippenhauer, and M. Ochoa, "Legacy-Compliant Data Authentication for Industrial Control System Traffic," in *Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS)*. Springer, 2017.

[4] O. Eikemeier, M. Fischlin, J.-F. Götzmann, A. Lehmann, D. Schröder, P. Schröder, and D. Wagner, "History-Free Aggregate Message Authentication Codes," in *Proceedings of the International Conference on Security and Cryptography for Networks (SCN)*. Springer, 2010.

[5] S. Eswaran, J. Edwards, A. Misra, and T. F. L. Porta, "Adaptive In-Network Processing for Bandwidth and Energy Constrained Mission-Oriented Multihop Wireless Networks," *IEEE Transactions on Mobile Computing*, vol. 11, no. 9, 2012.

[6] Florian Euchner, Felix Fellhauer, Marx Gauger, "Sigfox Radio Protocol Overview and Specifications," Tech. Rep., 2018.

[7] P. Ginzboorg, V. Niemi, and J. Ott, "Authentication of fragments with short tags," *Theoretical Computer Science*, 2024.

[8] T. Hänel, L. Brüggemann, F. Loske, and N. Aschenbruck, "Long-Term Wireless Sensor Network Deployments in Industry and Office Scenarios," in *Proceedings of the 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2021.

[9] S. Hirose and J. Shikata, "Non-adaptive Group-Testing Aggregate MAC Scheme," in *Proceedings of the International Conference on Information Security Practice and Experience (ISPEC)*. Springer, 2018.

[10] Y.-w. Hong, W.-j. Huang, F.-h. Chiu, and C.-c. J. Kuo, "Cooperative Communications in Resource-Constrained Wireless Networks," *IEEE Signal Processing Magazine*, vol. 24, no. 3, pp. 47–57, 2007.

[11] J. Katz and A. Y. Lindell, "Aggregate Message Authentication Codes," in *Proceedings of the Cryptographers' Track at the RSA Conference (CT-RSA)*. Springer, 2008.

[12] K. L. Ketshabetswe, A. M. Zungeru, B. Mtengi, C. K. Lebekwe, and S. R. S. Prabaharan, "Data Compression Algorithms for Wireless Sensor Networks: A Review and Comparison," *IEEE Access*, vol. 9, pp. 136 872–136 891, 2021.

[13] V. Kolesnikov, "MAC Aggregation with Message Multiplicity," in *International Conference on Security and Cryptography for Networks (SCN)*. Springer, 2012.

[14] V. Lesi, I. Jovanov, and M. Pajic, "Integrating Security in Resource-Constrained Cyber-Physical Systems," *ACM Trans. Cyber-Phys. Syst.*, vol. 4, no. 3, May 2020.

[15] H. Li, V. Kumar, J.-M. Park, and Y. Yang, "Cumulative Message Authentication Codes for Resource-Constrained IoT Networks," *IEEE Internet of Things Journal*, 2021.

[16] LoRa Alliance, "LoRaWAN™ 1.1 Specification," Tech. Rep., 2017.

[17] Morris Dworkin, "NIST SP 800-38B - Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication," Tech. Rep., 2016.

[18] National Institute of Standards and Technology, "Recommendation for Key Management: Part 1 – General," *NIST Special Publication 800-57 Part 1 Revision 5*, 2020.

[19] D. K. Nilsson, U. E. Larson, and E. Jonsson, "Efficient In-Vehicle Delayed Data Authentication Based on Compound Message Authentication Codes," in *Proceedings of the 68th Vehicular Technology Conference (VTC-Fall)*. IEEE, 2008.

[20] S. Pushpalatha and K. Shivaprakasha, "Energy-efficient communication using data aggregation and data compression techniques in wireless sensor networks: A survey," in *Advances in Communication, Signal Processing, VLSI, and Embedded Systems: Select Proceedings of VSPICE 2019*. Springer, 2019, pp. 161–179.

[21] S. Raza, D. Trabalza, and T. Voigt, "6LoWPAN Compressed DTLS for CoAP," in *Proceedings of the 8th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2012.

[22] E. Rescorla, R. Barnes, H. Tschofenig, and B. Schwartz, "Compact TLS 1.3," IETF, Internet-Draft, 2023.

[23] E. Rescorla, H. Tschofenig, and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3," IETF, RFC 9147, 2022.

[24] E. Rozner, J. Seshadri, Y. Mehta, and L. Qiu, "SOAR: Simple Opportunistic Adaptive Routing Protocol for Wireless Mesh Networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 12, pp. 1622–1635, 2009.

[25] J. Schmandt, A. T. Sherman, and N. Banerjee, "Mini-MAC: Raising the bar for vehicular security with a lightweight message authentication protocol," *Vehicular Communications*, vol. 9, 2017.

[26] A. Seferagić, J. Famaey, E. De Poorter, and J. Hoebeke, "Survey on Wireless Technology Trade-Offs for the Industrial Internet of Things," *Sensors*, vol. 20, no. 2, 2020.

[27] M. Serror, S. Hack, M. Henze, M. Schuba, and K. Wehrle, "Challenges and Opportunities in Securing the Industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, 2021.

[28] M. Serror, S. Vaaßen, K. Wehrle, and J. Gross, "Practical Evaluation of Cooperative Communication for Ultra-Reliability and Low-Latency," in *Proceedings of the 19th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2018.

[29] F. K. Shaikh and S. Zeadally, "Energy harvesting in wireless sensor networks: A comprehensive review," *Renewable and Sustainable Energy Reviews*, vol. 55, 2016.

[30] C. Szilagyi and P. Koopman, "Flexible Multicast Authentication for Time-Triggered Embedded Control Network Applications," in *Proceedings of the IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*. IEEE, 2009.

[31] E. Uysal-Biyikoglu, B. Prabhakar, and A. El Gamal, "Energy-Efficient Packet Transmission Over a Wireless Link," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 487–499, 2002.

[32] S. Vitturi, C. Zunino, and T. Sauter, "Industrial Communication Systems and Their Future Challenges: Next-Generation Ethernet, IIoT, and 5G," *Proceedings of the IEEE*, vol. 107, no. 6, 2019.

[33] E. Wagner, J. Bauer, and M. Henze, "Take a Bite of the Reality Sandwich: Revisiting the Security of Progressive Message Authentication Codes," in *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, 2022.

[34] E. Wagner, M. Serror, K. Wehrle, and M. Henze, "When and How to Aggregate Message Authentication Codes on Lossy Channels?" in *Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS)*. Springer, 2024.