Efficient Message Authentication in the Industrial Internet of Things

Von der Fakultät für Informatik der RWTH Aachen University
zur Erlangung des akademischen Grades

eines Doktors der Naturwissenschaften

genehmigte Dissertation

vorgelegt von

Eric Michel Wagner, M.Sc.

Berichter:

Prof. Dr.-Ing. Klaus Wehrle

Prof. Dr.-Ing. Lars Wolf

Tag der mündlichen Prüfung: 22.04.2026

Abstract

The accelerating integration of computing and networking into industrial networks and critical infrastructure is driving rapid digitalization. While this shift enables transformative capabilities like remote monitoring and interconnected manufacturing, it also poses a significant cybersecurity risk. Expanding connectivity paired with increased interest from sophisticated adversaries has led to a surge in cyberattacks, risking equipment damage and human safety. To defend against these threats, we must overcome the stringent constraints of Industrial IoT (IIoT) devices, which often lack the resources for traditional cybersecurity measures.

Critically, the IIoT must be protected by efficient message authentication to prevent data manipulations that could directly impact the physical environment. This dissertation addresses explicitly how to mitigate the overhead associated with message authentication, typically implemented using MAC schemes. A MAC is a cryptographic tag derived from a secret key and a message, appended to the message itself. Upon reception, the tag allows the recipient to verify that the message originates from the claimed source and has not been tampered with in transit.

In the IIoT, where messages are often only a few bytes long, these, typically 16 byte long, tags add significant per-message overhead. To reduce this overhead, we design the SP-MAC scheme with optimal resilience to induced or random packet loss. SP-MAC allows for optimistic processing of incoming data with initially reduced security, which strengthens as subsequent messages are received. Alternatively, SP-MAC can be used as an aggregated MAC scheme, focusing on whether and when a message achieves full security while still aggregating authentication over multiple messages. We demonstrate that MAC aggregation, not only with SP-MAC, is feasible over lossy channels and can lead to a goodput increase of up to 50 % in DTLS 1.3.

MACs also challenge the IIoT through processing overhead and latency overhead. To this end, we design RePeL, a procedure to retrofit security into legacy systems by embedding authentication data into unused protocol fields. IIoT devices can perform this embedding natively or offload it to Bump-in-the-Wire devices to reduce computational load. Complementing these efforts, we introduce BP-MAC, a lightweight MAC scheme optimized for low-latency authentication of short IIoT messages.

Finally, we investigate how MAC-based message authentication can be applied to group communication, where usually expensive digital signatures would be used. With Caiba, we leverage the low bandwidth of bus communication systems to enable a dedicated authenticator node to overwrite authentication tags during transmission. By splitting tag verification between the authenticator and receivers, the latter are prevented from impersonating the sender, as they lack information to generate valid tags. Additionally, we introduce Madtls, a protocol that embeds middleboxes with least-privilege data access into end-to-end secured IIoT connections. Thus, end-to-end security can be deployed without sacrificing the performance benefits of middleboxes, which are often critical in IIoT environments.

Together, these contributions demonstrate that efficient message authentication in resource-constrained IIoT environments is achievable through efficient resource utilization and scenario-specific optimizations. These findings pave the way for more secure, scalable, and resilient IIoT deployments in the face of evolving cyber threats.

Kurzfassung

Steigende Digitalisierung in der Industrie und kritischer Infrastrukturen ermöglicht transformative Funktionen wie Fernüberwachung, birgt jedoch auch erhebliche Risiken. Die zunehmende Vernetzung, in Verbindung mit dem gestiegenen Interesse einflussreicher Widersacher, hat zu einer Zunahme von Cyberangriffen geführt, welche Geräteschäden und Risiken für Menschenleben zur Folge haben. Dem Schutz gegen diese Bedrohungen stehen hierbei oft die knappen Ressourcen von IIoT-Geräte im Weg, die herkömmliche Cybersicherheitsmaßnahmen behindern.

Entscheidend ist die Verwendung einer effizienten Nachrichtenauthentifizierung zur Verhinderung von Datenmanipulationen mit direkter Auswirkung auf die physische Umgebung. Wir befassen uns dementsprechend explizit mit der Frage, wie diese Nachrichtenauthentifizierung, in der Regel durch MAC-Verfahren umgesetzt, an die vorhandenen Einschränkungen angepasst werden kann. Eine MAC wird aus einem geheimen Schlüssel und einer Nachricht abgeleitet und an die Nachricht selbst angehangen. Der Empfänger kann anhand dieser MAC anschließend den Ursprung und die Integrität der erhaltenen Daten überprüfen.

Bei oft nur wenigen Byte langen Nachrichten verursachen diese, typischerweise 16 Byte langen, MACs einen erheblichen Overhead. Zur Reduktion dieses Overheads haben wird das SP-MAC Verfahren mit optimaler Resilienz gegen Paketverlust entwickelt. SP-MAC ermöglicht eine optimistische Verarbeitung eingehender Daten mit zunächst reduziertem Schutz, welcher durch nachfolgender Nachrichten verstärkt wird. Alternativ kann SP-MAC als MAC-Aggregationverfahren verwendet werden, wobei nur wichtig ist ob, und wann vollständige Sicherheit erreicht wird. Wir zeigen, dass die Verwendung von MAC-Aggregation, u. a. durch SP-MAC, über verlustbehaftete Kanäle zu einer Goodput-Steigerung von bis zu 50 % in DTLS 1.3 führen kann.

Auch die erhöhte Verarbeitungszeit von MAC-Verfahren stellt eine Herausforderung dar. Um dem entgegenzuwirken, haben wir mit RePeL ein Verfahren zur Nachrüstung von Sicherheit in Legacy-Systemen entwickelt, welche die Einbettung von Authentifizierungsdaten in ungenutzte Protokollfelder ermöglicht. IIoT-Geräte können diese Einbettung nativ durchführen oder, zur Reduzierung der Rechenlast, an Bump-in-the-Wire Geräte auslagern. Ergänzend stellen wir BP-MAC als neues MAC-Verfahren vor, welches speziell für kurze IIoT-Nachrichten optimiert ist.

Schließlich untersuchen wir, wie MACs bei Gruppenkommunikation verwendet werden können, um teuere digitale Signaturen zu vermeiden. Mit Caiba nutzen wir die geringe Bandbreite von Bussen, um einem dedizierten Authentifizierer zu ermöglichen, MACs während der Übertragung zu überschreiben. Durch die Aufteilung der Verifizierung zwischen dem Authentifizierer und den Empfängern wird garantiert, dass letztere nie alle Informationen zur Generierung gültiger MACs haben. Zusätzlich führen wir das Madtls Protokoll ein, um Middleboxen mit *least-privilege* Datenzugriff in Ende-zu-Ende gesicherte IIoT-Verbindungen einzubetten. Auf diese Weise muss beim Einführen von Sicherheitsmaßnahmen nicht auf wichtigen Performanzgewinne, die durch Middleboxen ermöglicht werden, verzichtet werden.

Diese Beiträge zeigen, dass eine wirksame Nachrichtenauthentifizierung durch effiziente Ressourcennutzung und szenariospezifische Optimierungen erreicht werden kann. Damit ebnen wir den Weg für ein sicherere, skalierbarere und widerstandsfähigere Netze trotz sich ständig weiterentwickelnder Cyber-Bedrohungen.

Acknowledgments

First, I want to thank the people with whom this journey began. All started with Jens Hiller, who offered me my first position as a student research assistant during my Bachelor's degree and sparked my enthusiasm for research. I would also like to express my gratitude to Martin Serror and Roman Matzutt, with whom I continued to collaborate on various projects throughout my studies.

Second, I want to expand my gratitude to the CA&D team at Fraunhofer FKIE (Martin Henze, Elmar Padilla, Martin Serror, Lennart Bader, Jan Bauer, Konrad Wolsing, Luisa Lux, and, Frederik Basels) for providing such a pleasant and supportive research environment. I was always given the freedom to explore my own ideas, even when they leaned more toward fundamental research than Fraunhofer typcial research areas. Thank you for your trust and support.

Third, I would like to thank the people at COMSYS, especially the members of the S&P group (Jan Pennekamp, Jens Hiller Roman Matzutt, Markus Dahlmanns, Ina Fink, Christian van Sloun, Johannes Lohmöller) who welcomed Konrad and me as external Ph.D. students. Their openness in sharing knowledge and experience, as well as their guidance throughout my doctoral journey, greatly contributed to the successful completion of this thesis and my defense.

Fourth, I would like to thank all the students whom I had the pleasure to supervise (Luisa Lux, Frederik Basels, Nils Rothaug, Nils Kattenbeck, Lasse Moench, Dominik Kus, Patrick Wagner, Christophe Haag, Irakli Bajelidze, David Heye, Mazen Al Aswad, and Jonas Kocszwara). Their contributions, ideas, and dedication were invaluable, and this thesis would not have been possible without them.

Fifth, I want to thank Klaus Wehrle, Lars Wolf, Redha Gouicem, and Ulrike Meyer for agreeing to be be part of my Ph.D. defense committee.

Finally, I would like to thank everyone who supported and accompanied me throughout this long and exciting journey, my family, friends, colleagues, and collaborators. Your encouragement and support meant more to me than words can express.

Published Papers

Parts of this thesis are based on the following peer-reviewed papers that have already been published. All my collaborators are among my co-authors. A detailed attribution of contributions can be found on the following page.

List of Publications

[WBB+25] Eric Wagner, Frederik Basels, Jan Bauer, Till Zimmermann, Klaus Wehrle, and Martin Henze. CAIBA: Multicast Source Authentication for CAN Through Reactive Bit Flipping. In *Proceedings of the 2025 IEEE 10th European Symposium on Security and Privacy (EuroS&P'25)*, 2025.

[WBH22] Eric Wagner, Jan Bauer, and Martin 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 (WiSec'22)*, 2022.

[WHS+24] Eric Wagner, David Heye, Martin Serror, Ike Kunze, Klaus Wehrle, and Martin Henze. Madtls: Fine-grained Middlebox-aware End-to-end Security for Industrial Communication. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security (AsiaCCS'24)*, 2024.

[WHSW25] Eric Wagner, David Heye, Martin Serror, and Klaus Wehrle. Integrating MAC Aggregation over Lossy Channels in DTLS 1.3. In *Proceedings of the 33rd IEEE International Conference on Network Protocols (ICNP'25)*, 2025.

[WRW+23] Eric Wagner, Nils Rothaug, Konrad Wolsing, Lennart Bader, Klaus Wehrle, and Martin Henze. Retrofitting Integrity Protection into Unused Header Fields of Legacy Industrial Protocols. In *Proceedings of the IEEE 48th Conference on Local Computer Networks (LCN'23)*, 2023.

[WSWH22] Eric Wagner, Martin Serror, Klaus Wehrle, and Martin Henze. BP-MAC: Fast Authentication for Short Messages. In *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec'22)*, 2022.

[WSWH24] Eric Wagner, Martin Serror, Klaus Wehrle, and Martin 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'24)*, 2024.

Attributions of Contributions

Most contributions presented in this dissertation are based on ideas by the author of this thesis that have been refined and implemented in collaboration with my students in the context of their Bachelor's thesis, Master's thesis, or student assistant positions as well as my colleagues. The resulting publications, that were created with the support of the respective co-authors, provide the foundation of this dissertation. If not stated otherwise, the author of this dissertation is responsible for the initial research ideas, the design of proposed solutions, the implementations, the conceptualization and conduction of evaluations, the security proofs, as well as their final publication.

SP-MAC [WBH22]. Martin Henze assisted in refining the research questions surrounding the idea of adapting progressive message authentication to wireless and thus lossy channels. Misha Lavrov pointed us at the concept of Golomb Rulers that the author of this thesis proved to yield optimal resilience to packet loss for progressive message authentication.

MAC Aggregation on Wireless Channels [WSWH24, WHSW25]. First simulation results for the investigation MAC aggregation on wireless channels were produced by Nils Kattenbeck in the context of his Bachelor's thesis. The author of this thesis improved and refined these simulation results. The design of the extension to integrate MAC aggregation into DTLS 1.3 were done in collarboration with David Heye in the context of his Master's thesis. David Heye also implemented this extension and assisted in conducting the practical evaluation of the scheme.

RePeL [WRW+23]. The initial idea to investigate to which degree unused protocol fields in industrial protocols can be used as additional space for authentication tags stems from Martin Henze. In the context of his Bachelor's thesis, Nils Rothaug implemented the *RePeL* library and assisted with the final evaluation of the system.

CAIBA [WBB+25]. The CAIBA protocol has been designed in collaboration with Frederik Basels, who also implemented and evaluated the protocol under my supervision in the context of his Master's thesis. Till Zimmermann assisted us with his electrical engineering knowledge especially in conjuncture with the CAN protocol.

MADTLS [WHS+24]. Ike Kunze contributed with potential use cases of access-restricted middleboxes in industrial networks. The MADTLS protocol was implemented and evaluated under my supervision by David Heye in the context his Bachelor's thesis.

Contents

1

> *Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning.*

Winston Churchill, *1942*

Introduction

The accelerating pace of digitalization leads to an unprecedented integration of computing and connectivity into everyday objects and industrial systems. At the forefront of this transformation are an increasing number of tiny computers, so-called Internet of Things (IoT) devices, whose number was projected to surpass 75 billion in 2025 [215]. Most people have heard of the IoT in the context of smart home products such as networked light fixtures, thermostats, or other appliances [185]. Nowadays, the emergence of the Industrial IoT (IIoT), however, revolutionizes many more sectors [227]. IIoT devices are thus deployed or envisaged for deployment in a wide range of industrial and critical infrastructure scenarios, ranging from industrial automation [220] to underwater environmental monitoring [243].

The increased connectivity offered by the IIoT introduces new capabilities such as interconnected manufacturing plants or remote control [16]. At the same time, the increasing ubiquity of IIoT devices [227], unfortunately, also draws attention from cyber criminals [16]. Especially applications managing strategic and safety-critical information pose as an attractive target not only for individual malicious actors but also for state-sponsored adversaries [207]. And while the security of, for example, manufacturing plants originally relied on being "air-gapped" systems [107], this defense layer no longer offers sufficient protection against advanced technical and social attack techniques [16]. The accompanying rise in potentially vulnerable or infected IIoT devices massively inflates this cybersecurity risk, especially considering that many IIoT devices rely on wireless, and thus broadcasted, communication [56].

However, unlike traditional computing systems, IIoT devices often lack the computational and energy resources to support conventional security mechanisms [148]. As these devices often operate under severe resource constraints, traditional cryptographic protocols and security frameworks prove infeasible or inefficient [148]. For example, protecting the secrecy and authenticity of IIoT traffic with cipher suites such as `AES CCM` on constrained processors such as the MSP430 requires several milliseconds of computations, which exceeds tight latency bounds commonly observed [101].

These constraints make it challenging to ensure the authenticity, integrity, and secrecy of communication channels without compromising the performance or lifespan of the devices. Consequently, developing lightweight yet effective security solutions tailored for such environments has emerged as a pressing research priority [148].

Despite ongoing advancements, there remains a critical gap in security solutions tailored to resource-constrained devices. Recent research has focused on efficient encryption for data confidentiality [37] or process-aware intrusion detection [133]. Meanwhile, message authentication is essential to prevent the manipulation of sensor readings or control commands in the IIoT. Such adversarial tampering can induce equipment damage or even endanger human lives due to the cyber-physical nature of these systems. To address these risks, this dissertation investigates approaches for adapting message authentication to the stringent constraints of the IIoT.

1.1 Problem Analysis

To understand the current state of research on securing the IIoT and resulting gaps, we first take a step back and look at how connections between powerful devices are traditionally secured. In general, more powerful modern devices, *e.g.*, smartphones, have enough resources that the overhead of modern security protocols becomes negligible. At the same time, the cryptographic primitives underlying these protocols (*e.g.*, HMAC or ECDSA) have a proven track record of withstanding extensive efforts to break them. Thus, most vulnerabilities nowadays stem from misconfigurations [161], social engineering [166], and protocol or implementation level vulnerabilities [42]. However, IIoT devices do not have the luxury to rely on these powerful cryptographic primitives due to resource limitations. Meanwhile, relying on more lightweight primitives introduces a significant risk, as these primitives have not been analyzed with the same scrutiny as the widespread and established mechanisms. In this dissertation, we therefore tackle the challenge of extending the strength of well-established cryptographic primitives to resource-constrained environments where their traditional application may consume too many resources.

1.1.1 The Three Pillars of Secure Communication Channels

To conduct a holistic discussion on securing the IIoT, we must first clarify what security entails. Therefore, we must consider the three foundational pillars of information security: Confidentiality, Integrity, and Availability [174]. These properties each represent a unique role in safeguarding communication channels against abuse, as detailed below.

Confidentiality. Confidentiality refers to the protection of sensitive data against access by unauthorized parties. For in-transit data of a communication channel, the usual means to achieve confidentiality is encryption based on a symmetric key exchanged in advance between the communicating parties. Other techniques to achieve confidentiality, such as covert channels (*e.g.*, in TCP/IP headers [168]), are

less reliable and risk being decoded. Many lightweight encryption algorithms have been proposed over the years [37], and even the common AES algorithm performs well on embedded devices, especially as it can be efficiently implemented in hardware [37].

Availability. Availability refers to the data being available to the entities requiring them to operate, *i.e.*, the availability of the communication channels to reach the intended receivers. In contrast to confidentiality and integrity, availability cannot be achieved with cryptographic tools but rather through a resilient protocol and system design and (physical) protection from *e.g.*, jamming attacks that could disrupt a communication channel. Still, all efforts to improve confidentiality and integrity must consider their effects on availability, especially in industrial networks and critical infrastructure.

Integrity. Data integrity ensures that received messages match those circulated by the sender. On a message or packet-based communication channel, the means to achieve data integrity is message authentication. Thereby, message authentication fulfills two roles: It ensures that messages stem from the alleged trustworthy source and that messages have not been tampered with during transmission. Without message authentication, an adversary could alter the data, even if the data were encrypted, and they thus do not know its content and what they alter it to. To realize integrity protection, different cryptographic schemes can be employed, such as digital signatures or Message Authentication Codes (MACs).

The priorities of these properties differ between the IIoT and the classical Internet. While an e-commerce operator would rather be offline than leak its clients' banking data, a power grid operator's primary goal is reliable control over its power grid, even if it means that communication is sent in plaintext. For the IIoT, it is thus crucial to provide efficient means for message authentication.

1.1.2 Paths Towards Message Authentication

Message authentication introduces fixed space and processing overhead that is relatively independent of message sizes. While this overhead is often negligible in traditional computing environments, it becomes particularly pronounced in resource-constrained IIoT applications, where individual data transmissions may consist of only a few bytes [251]. This phenomenon may partially explain why optimizing message authentication has received comparatively little attention, as drawbacks only materialize in constrained environments. On the other hand, this fixed overhead quickly becomes the dominant contributor to overall security-related costs in IIoT scenarios. Given the critical role of integrity protection and the scarcity of targeted research, there is a pressing need to adapt message authentication mechanisms to the unique constraints of the IIoT. This dissertation addresses that need by focusing on improving the efficiency of message authentication both at the device level and within the network.

To do so, we begin by examining the foundational approaches to cryptographically proven message authenticity and the challenges they present in constrained settings. Broadly, message authentication can be achieved using either symmetric or asymmetric cryptographic primitives [39].

Symmetric Cryptography. Symmetric cryptography is based on a common secret key shared among participants in a communication, *i.e.*, the sender and the receiver(s). For the goal of message authentication, this secret key, along with the payload, is used to compute authentication tags, so-called MACs, that are integrated into a Protocol Data Unit (PDU). To meet the generally recommended 128-bit security levels [173], each message must information-theoretically be protected by a cryptographically secure tag of at least 16 bytes. Receivers knowing the secret key can verify the authentication tag. If a third party, not knowing the secret key, tries to manipulate a PDU, it cannot create a valid authentication tag and, hence, the manipulation can be detected.

Asymmetric Cryptography. With asymmetric cryptography, the keys to authenticate and verify a message are different. In this case, senders use a private key, only known to them, to generate a digital signature, which is again integrated into a PDU. To verify this signature, a receiver then only needs to know the public key linked to the private key. Digital signatures thus allow for multiple entities to verify a message without giving them the ability to impersonate the sender. However, this comes at a cost. Digital signatures are significantly longer and require more computations than MACs to achieve the same security level.

Generally, MACs offer superior efficiency in terms of computational resources and bandwidth usage compared to digital signatures, making them the natural choice to provide message authentication in the IIoT [41]. However, even MAC schemes often fall short of the efficiency required in highly constrained IIoT environments [25]. To understand the challenges of deploying message authentication in the IIoT, we must first study the intricacies of its diverse constraints.

1.2 Resource Constraints in the Industrial IoT

The IIoT encapsulates a vast variety of applications, ranging from industrial automation over environmental monitoring to even underwater communication, such that establishing general requirements toward the demands of security mechanisms is impossibly restrictive. Instead, in Section 1.2.1, we first provide a taxonomy of the different dimensions of resource constraints potentially encountered in the IIoT. Afterward, in Section 1.2.2, we take a look at three concrete example scenarios to illustrate the vastly different demands of each one of them.

1.2.1 A Taxonomy on Resource Constraints in the IIoT

Looking at the range of potential resource constraints in the IIoT, we observe a very diverse landscape. Potential constraints can, for example, arise from strict latency requirements, *e.g.*, in Industrial Control Systems (ICSs) [155], or from low bandwidth capacities, *e.g.*, in long-range acoustic underwater communication [144]. In the following, we provide a taxonomy of the most common constraints in the IIoT based on the challenges identified in RFC 9556 [105]. We discuss and illustrate in

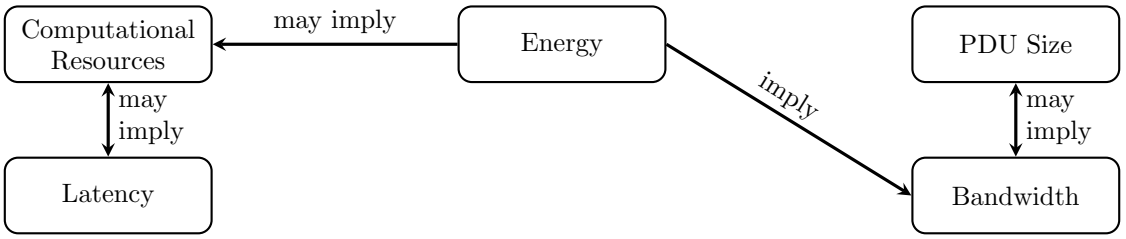


Figure 1.1 IIoT scenarios can be influenced by different constraints that restrict the deployment of adequate cybersecurity. Some constraints can imply the existence of other constraints, but in general, different scenarios can be exposed to different subsets of these constraints.

Figure 1.1 how these constraints interact with each other. These interactions show how some inherent constraints of a scenario lead to other constraints, but also suggest that not every scenario is affected by each constraint. We confirm this hypothesis in the subsequent section by looking at concrete example scenarios. Ultimately, this observation opens the door to designing optimized trade-offs between these constraints to adapt security mechanisms to concrete demands.

Bandwidth. In many IIoT scenarios [20], bandwidth is limited, either because the underlying communication only supports low-bandwidth communication (*e.g.*, LoRaWAN) or because the communication channel is shared by many devices (*e.g.*, industrial automation). Scenarios that expose bandwidth limitations may result in limited lengths of PDUs due to requirements by the used protocols across the ISO/OSI reference stack or because update frequency must remain high. In other cases, aggregating data into a single PDU can alleviate bandwidth constraints because header overhead is reduced.

Computational Resources. Some IIoT devices operate with especially low processing power [101], memory, and storage to save costs or reduce the devices' physical footprint. In these situations, the computation of (non-optimized) cryptographic functions can consume a substantial amount of a device's resources and thus also delay communications. Such limitations may restrict the deployment of security measures in the field.

Energy. While some IIoT devices may be connected to a power source, *e.g.*, in a manufacturing environment, others are battery-operated, such as remotely deployed environmental monitoring nodes [135]. These devices must consider the energy cost of any security measures, as they can greatly impact their lifetime. Here, the wireless transmission of data is the main factor because, in many cases, its power consumption is an order of magnitude higher than the power consumption of processing [222]. Thus, an energy-constrained IIoT device benefits significantly from reducing data transmissions and can also marginally benefit from reducing computations.

PDU Size. Mostly due to protocol constraints in *e.g.*, LoRaWAN or CAN, individual packets may be heavily constrained in their size. LoRaWAN, for example, reserves only 32 bit for integrity protection in its packets [151]. These design choices were made to increase the reliability of long-range communication and allow everyone to gain access to the transmission medium. However, they oppose the deployment of traditional security measures, *e.g.*, a 16-byte MAC.

Constraints	Industry 4.0	Environmental Monitoring	Underwater Acoustic Comm.
Bandwidth	○	●	●
Comp. Resources	●	-	-
Energy	-	●	-
PDU Size	○	●	●
Latency	●	-	-
		● full	○ partial

Table 1.1 IIoT scenarios expose vastly different constraints toward that must be considered for the design of a cybersecurity solution.

Latency. Mission-critical machine-to-machine communication in IIoT scenarios often requires low-latency networking [155]. The corresponding messages span safety-critical shutdown commands to the precise operation of production lines [101]. Depending on the available computational resources, the cryptographic processing may consume valuable time, or the deployment of more efficient algorithms with higher memory footprints may be prevented.

We observe that the constraints in IIoT networks can take different forms that cluster around either the necessary computational resources or the required bandwidth. While one constraint can imply the existence of others, *e.g.*, energy-constrained scenarios imply that channel utilization must be reduced, the existence of most constraints is mainly dependent on the concrete scenario.

1.2.2 Example Scenarios and their Resource Constraints

In the following, we analyze three scenarios to study how resource constraints actually limit cybersecurity deployments in practice. As summarized in Table 1.1, we find that different scenarios expose unique combinations of constrained resources, while other resources are available in excess.

Industry 4.0. Industry 4.0 describes the vision of creating a highly automated and interconnected manufacturing environment. To realize this vision, reliable (wireless) communication with delays of a few milliseconds and below is required [82, 155]. Factory automation, *i.e.*, the automation of production lines, requires data update rates of up to 1 kHz. Meanwhile, up to 100 nodes are involved in a single production line, leading to bandwidth requirements of multiple MB per second [155]. Considering the overheads of short packets and sharing the transmission medium between multiple devices, Industry 4.0 scenarios thus expose bandwidth constraints. Additionally, the deployed IIoT devices often have limited computational resources to reduce their size and save costs [270]. On the other hand, energy is usually readily available on a factory floor and thus does not limit the deployment of cybersecurity measures.

Environmental Monitoring. The monitoring of, *e.g.*, pollution levels, has vastly different requirements for its hardware and network than the Industry 4.0. Here, the primary constraint is battery lifetime, as devices must be able to operate autonomously for multiple years [135]. As a direct consequence, the amount of communicated data must be minimized to conserve energy, and many nodes share the same long-range transmission channels. Meanwhile, the communication latency has little importance, with delay of multiple hours being often acceptable [135].

Underwater Acoustic Communication. Underwater communication primarily relies on acoustic channels due to the strong attenuation of electromagnetic signals. The acoustic channels offer long-range communication over multiple kilometers, but with low propagation speeds and low bandwidth. Underwater acoustic protocols, therefore, have very short PDUs. For example, the maximum PDU size of the GUWMANET [85] protocol is 128 bit. Hence, PDU sizes and bandwidth are heavily restricted, while long propagation delays make processing overhead negligible.

These scenarios illustrate the difference between constraints in IIoT scenarios. While a one-fits-all solution to bring cybersecurity into this landscape is desirable, it is also unrealistic to develop solutions that address all constraints at once. Instead, we must identify the right trade-offs to optimize message authentication for specific scenarios.

1.3 Key Observations and Research Questions

We identified MACs schemes as the most promising candidate solution for efficient message authentication in the IIoT. However, today's MAC schemes often lack the efficiency required to operate within the diverse constraints of the IIoT [25]. These constraints primarily cluster around limited computational resources and restricted bandwidth. Since not all constraints impact every IIoT scenario equally, our research is guided by two key questions (RQ1 and RQ2), each addressing one of these constraint clusters. Additionally, we explore how to unlock the superior performance of MACs to communication scenarios with more than two parties that typically rely on heavyweight digital signatures (RQ3).

RQ1 – How can the bandwidth overhead of message authentication be optimized?

One challenge of message authentication in the IIoT is the size of authentication tags. In the classical Internet, where typical TLS records are several 100 byte long, the appending of a 16 byte authentication tag (for recommended 128-bit security [173]) is only a marginal overhead. However, with payloads that are only a few byte long [82, 155], this overhead becomes significant. In the past, the idea of aggregating the authentication of multiple PDUs was explored [115], however, neither in the context of latency-critical communication nor considering the possibility of wireless, and thus lossy, channels. In Chapter 3 of this dissertation, we explore this possibility and show how MAC aggregation can lead to significant bandwidth saving in real-world IIoT deployments. As a result, we design SP-MAC [246] as a message authentication scheme that allows the optimistic processing of messages with initially reduced security and integrate MAC aggregation into the DTLS 1.3 protocol [248, 251].

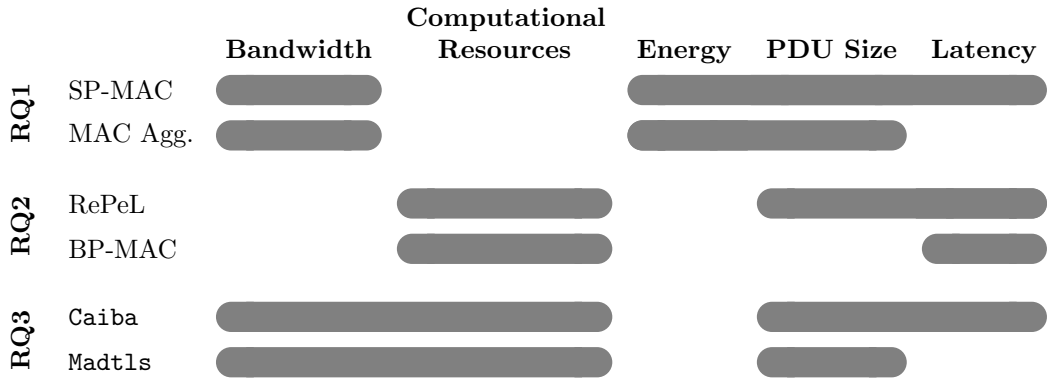


Figure 1.2 While none of the MAC schemes, protocols, and systems designed within this thesis MAC considers all major constraints, the individual solutions offer different trade-offs for a wide variety of the IIoT scenarios.

RQ2 – How can cryptographic processing and resulting delays of message authentication be minimized?

Cryptographic operations are comparatively expensive in terms of processing in relation to other stages of the networking stack. The low-power hardware in the IIoT increases the significance of this overhead. While current research proposes performance improvements for MAC schemes on resource-constrained hardware [21, 127, 165], it remains an open question how the fact that most critical messages in the IIoT are only a few bytes long can contribute to minimizing latency. In Chapter 4 of this dissertation, we therefore explore *(i)* how to efficiently embed authentication efficiently into the networking stack with RePeL [249] and *(ii)* the possibility of precomputing partial authentication tags during idling times to speed up latency-critical processing with BP-MAC [250].

RQ3 – How can the efficiency of symmetric cryptography be unlocked for group communication systems?

To ensure message authenticity if more than two parties are involved in a communication channel, *e.g.*, for broadcasting channels, digital signatures are usually deployed. They offer a clear differentiation between the generator and the verifier of a signature. However, the necessary asymmetric cryptography is less performant and occupies more space in a PDU than MACs, such that they are less applicable in the IIoT. In Chapter 5 of this dissertation, we explore how we can use MACs even in these scenarios to *(i)* involve access-restricted middleboxes in a communication channel through the Madtls [247] protocol and *(ii)* authenticate the source of broadcasted messages in bus-based communication with Caiba [245].

To answer these three research questions, we design novel MAC schemes, protocols, and systems that adopt message authentication to the constraints of the IIoT. Figure 1.2 showcases these advancements and the constraints that are considered either directly or indirectly for their design. While no MAC scheme, protocol, or system addresses all constraints, they each provide a unique solution to various scenarios with unique trade-offs.

2

> *If I have seen further, it is by standing on the shoulders of Giants.*
>
> Isaac Newton, *1675*

Background on Message Authentication Codes

In this dissertation, we investigate how message authentication can be adapted to the constraints of the Industrial IoT (IIoT). Efficient message authentication for in-transit data is generally achieved by Message Authentication Codes (MACs). However, even modern MAC schemes are challenged by the constraints of the IIoT. In the following chapter, we formally introduce MAC schemes, establish the notations used throughout this dissertation, and take a look at some relevant MAC constructions.

2.1 Definition of Message Authentication Codes

A MAC scheme $\Sigma = (\mathit{Sig}, \mathit{Vrfy})$ is composed of two algorithms, a signing algorithm *Sig* and a verification algorithm *Vrfy* [39]. The algorithms generate and verify so-called authentication tags $t \in \mathcal{T}$ of messages $m \in \mathcal{M}$ with the help of a pre-shared secret key $k \in \mathcal{K}$. The signing algorithm *Sig* allows the owner of the key k to generate a tag t for a message m:

$$\mathit{Sig}_k(m) = t$$

The verification algorithm *Vrfy*, on the other hand, allows the owner of the key k to verify the integrity of a *received* message-tag pair $(m^*, t^*) \in \mathcal{M} \times \mathcal{T}$:

$$\mathit{Vrfy}_k(m^*, t^*) = \begin{cases} \texttt{accept} & \text{if the tag } t^* \text{ is valid} \\ \texttt{reject} & \text{if the tag } t^* \text{ is invalid} \end{cases}$$

2.2 Properties of MACs

To provide secure message authentication, MAC schemes must be *sound* and *forgery-resistant* as detailed below. Moreover, most practical MAC schemes, and all those encountered in this dissertation, are deterministic.

Soundness. A MAC scheme is *sound* if each tag generated by the signing algorithm Sig_k is accepted by the verification algorithm $Vrfy_k$ (for identical keys k and messages m) [39].

Forgery-Resistance. A MAC scheme is *forgery-resistant* if there exists no efficient algorithm for an adversary not knowing the key k to generate a valid tag [39].

Deterministic. A MAC scheme is *deterministic* if the verification can be performed by recomputing the tag with the help of the signing algorithm [39]:

$$Vrfy_k(m^*, t^*) = \begin{cases} \texttt{accept} & \text{if } Sig_k(m^*) = t^* \\ \texttt{reject} & \text{otherwise} \end{cases}$$

2.3 Security of MAC Constructions

MAC schemes need to withstand powerful adversaries in order to protect the IIoT. The main security concerns MAC schemes face are tag forgeries [39], which we introduce in Sections 2.3.1. A MAC scheme is considered secure if it is resilient against such forgeries. In Section 2.3.2, we additionally discuss replay attacks, which are also a common concern in the context of integrity protection. However, the protection against such replay attacks is usually not part of the security definition of MAC schemes, as replayed messages would still be authentic by their definition. Nevertheless, MAC schemes can aid in mitigating replay attacks, and some MAC constructions even natively protect against replay attacks. Finally, in Section 2.3.3, we also introduce how the security level of secure MAC schemes can be quantified.

2.3.1 Tag Forgeries

In a forgery attack, an adversary attempts to efficiently generate one or multiple valid tags for previously unobserved messages. A MAC scheme is considered secure if it prevents such tag forgeries [39]. Formally, resistance to forgery attacks means that, even under a chosen-message attack, an adversary cannot create an *existential* tag forgery, *i.e.*, the attack cannot generate a tag t for a previously unseen message m, that $Vrfy_k(m, t)$ would accept. This security property of a MAC scheme Σ is typically defined through a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. In this game, the adversary $\mathcal{A}$ tries to learn patterns in the tag generation by requesting valid tags for messages from an oracle and subsequently guessing a valid tag for a new message.

Attack Game 1

- $\mathcal{C}$ randomly picks a key k from $\mathcal{K}$.
- $\mathcal{A}$ repeatedly queries an oracle with a message m_i for a valid tag t_i, *i.e.*, t_i such that $\text{Vrfy}_k(m_i, t_i)$ returns `accept`. Denote $\mathbb{M}$ the set of all m_i queried by $\mathcal{A}$.
- Eventually, $\mathcal{A}$ outputs a candidate forgery $(m, t) \in \mathbb{M} \times \mathcal{T}$, with $m \notin \mathbb{M}$.

$\mathcal{A}$ wins the above game if $\text{Vrfy}_k(m, t)$ returns `accept`. We define $\mathcal{A}$'s advantage with respect to Σ, denoted as $\text{Adv}_{\mathcal{A}}[\Sigma]$, as the probability that $\mathcal{A}$ wins the game. A MAC scheme is considered secure if the advantage of all efficient adversaries $\mathcal{A}$ with respect to Σ is negligible, *i.e.*, $\text{Adv}_{\mathcal{A}}[\Sigma] = Pr[\mathcal{A} \text{ wins Attack Game 1 for } \Sigma] < \varepsilon$. For an *ideal* MAC scheme, $\mathcal{A}$ gains no advantage from querying the oracle and has no better strategy than guessing a tag uniformly at random.

Note that this security proof concerns the design of the MAC scheme Σ. Implementation errors or side channel attacks could still enable key recovery attacks, where the adversary gains direct access to the secret key k that should only be known by authorized signers and verifiers [42]. Once in possession of this key k, the adversary can generate valid tags for arbitrary messages, *i.e.*, *universal* forgeries, and thus completely undermine the security of the MAC scheme.

2.3.2 Replay Attacks

An attack may attempt to replay previously recorded messages at a later time when those same messages can trigger unintended behavior. In an Industrial Control System (ICS) scenario, for example, such replayed commands could move a robot arm to a then unsafe position. MAC schemes themselves do not protect against such attacks as they have no sense of time: A tag is either valid or invalid independent on when and by whom it is verified.

The mitigation of such replay attacks is thus usually delegated to the security layer protocol and not the MAC scheme itself. Mitigation is typically realized by ensuring the recency of received Protocol Data Units (PDUs), and not accepting the same PDU twice. In practice, the uniqueness of PDUs is usually ensured by incorporating nonces, often in the form of sequence numbers. Thus, a verifier can keep track of which PDU it received within a sliding window around the most recent sequence number and reject all older sequence numbers. Thus, while MAC schemes do not inherently protect against replay attacks, they aid in mitigating such attacks by ensuring that the sequence number and remaining data were linked by the sender.

2.3.3 Quantifying the Security of MAC Schemes

Against an ideal MAC scheme, an adversary without knowledge of the key may still guess a valid tag at random. In particular, a randomly guessed tag has a probability of 1 in $2^{|t|}$ of being validated by the verification algorithm, where $|t|$ stands for the

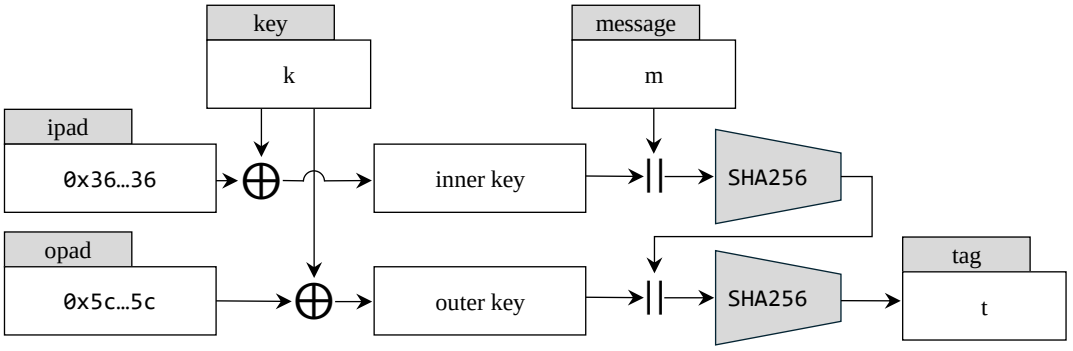


Figure 2.1 The HMAC construction can be used to convert cryptographic hash functions, such as SHA256 in this example, into MAC schemes.

length of the tag t in bits [171]. Hence, we can quantify the security of an ideal MAC scheme by the bit length of its tags. Longer tags thus provide more security but also consume more space in a PDU. Consequently, tags can be truncated to a desired length with a linear and controlled drop of the achieved security level.

General recommendations for the desirable security levels of authentication tags are hard to formulate due to various practical limitations that restrict reasonable achievable security levels, *e.g.*, Low-power Wide-area Network (LPWAN) protocols such as LoRa [151] and Sigfox [77] do not offer the PDU space for long tags. Still, the German Federal Office for Information Security (BSI) recommends a tag length of at least 128 bit for `HMAC-SHA256` [75], a popular MAC scheme that we take a closer look at in the following section.

2.4 MAC Constructions

Over the years, different MAC constructions have been proposed for the design of concrete MAC schemes. In the following, we give an overview of three such constructions that lead to some of the most widely used MAC schemes today. Each construction has its own benefits, such as efficiency by combining data encryption and tag computation.

2.4.1 HMAC

Hash-based Message Authentication Codes (HMACs) have already been standardized in 1997 [125] and are attractive as they convert any cryptographic hash function into a MAC scheme, thus facilitating their design. Thus, while initially considered hash functions, *e.g.*, MD5 [204], are no longer considered secure, HMACs, now commonly used in combination with `SHA256` as cryptographic hash function [70], are still popular today. This HMAC scheme is referred to as `HMAC-SHA256` and used, for example, in the IPSEC [116] and TLS 1.2 [65] protocols.

The HMAC construction is based on a double hashing as illustrated in Figure 2.1. First, the secret key k is padded with null bytes to the length of the chosen hash

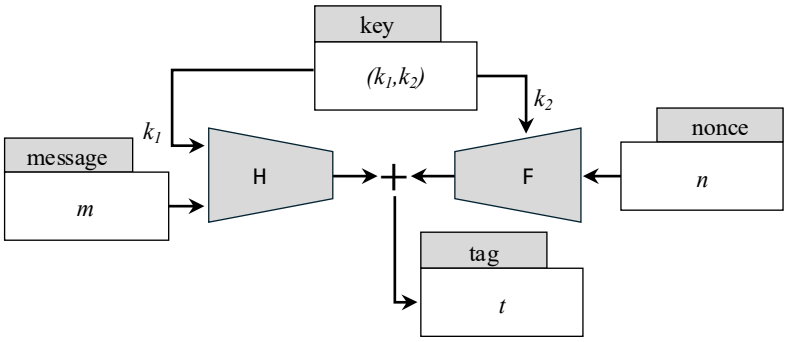


Figure 2.2 The Carter-Wegman MAC allows the construction of secure MAC schemes from a universal hash function H and a pseudorandom function F. It is the basis of many secure and efficient MAC schemes, such as UMAC and Poly1305.

function's block length, 64 byte in the case of `SHA256`. HMAC derives an inner key and an outer key from this secret key by XORing it with an inner pad (opad) and an outer pad (opad), respectively. The opad, consisting of `0x5c` bytes, and the ipad, consisting of `0x36` bytes, are of the same length as the secret key and chosen to minimize correlation between the inner and outer keys [30]. Then, the inner key is concatenated with the message m and hashed by the cryptographic hash function h, `SHA256` in our example. The outer key is concatenated with the digest of this computation and hashed again with the h. Finally, a `HMAC-SHA256` tag is computed by truncating the result of a layered hashing of the concatenated keys and messages to the leftmost 128 bit:

$$\texttt{HMAC-SHA256}_k(m) = \text{SHA256}((k \oplus \text{opad})||\text{SHA256}((k \oplus \text{ipad})||m))[:128]$$

This truncation reduces the information available to an adversary and shrinks the size of the transmitted tag [125]. Meanwhile, this truncation does not reduce the security of the `HMAC-SHA256` scheme, which is bound by the general birthday-based forgery attacks by Preneel and van Oorschot on iterated MAC schemes [29, 193].

2.4.2 The Carter-Wegman MAC Construction

Especially on embedded hardware, HMAC-based MAC schemes do not always provide sufficient performance. As an alternative, Carter-Wegman MAC constructions, such as UMAC [127] and Poly1305 [34], have recently gained popularity. Poly1305 is, for example, supported in the Internet Key Exchange (IKE) protocol [182], IPSEC [182], TLS 1.3 [200], and DTLS 1.3 [202].

Carter-Wegman MAC constructions add an additional nonce n to the parameters of Sig_k and $Vrfy_k$ [39], such that they natively protect against replay attacks. They boost efficiency by splitting tag computations, used for signing and verifying tags, into two steps, as shown in Figure 2.2. The main performance benefits stem from applying the relatively expensive cryptographic function only on the short n, while the

arbitrarily long message m is hashed by a potentially much more efficient *universal*, but not necessarily cryptographic, hash function H.

Therefore, the secret key k for Carter-Wegman MAC schemes is also split into two parts, k_1 and k_2. In a first step, the message m is hashed by the universal hash function H. The universal hash function is a secret hash function for which it is hard to find a collision as long as the adversary does not know a single output of H [39]. Moreover, Bernstein proved strong security bounds if H additionally is *difference unpredictable*, *i.e.*, that the probability $Pr[H(m') = H(m) + g]$ for a fixed number g is small [33]. To remain secret, the function H is pseudorandomly selected from a family of hash functions with k_1. Secondly, the comparably short nonce n is converted into a pseudorandom mask, or *blinding tag*, of the same length as the output of H with the help of a pseudorandom permutation F, keyed with k_2. The tag is then computed by adding the digests of H and F:

$$Sig_k(m, n) = H_{k_1}(m) + F_{k_2}(n)$$

The security of Carter-Wegman MAC schemes relies on the underlying primitives. Concretely, a Carter-Wegman MAC is provably secure if H fulfills the universality and difference unpredictability properties and if F is a pseudorandom function.

2.4.3 MACs in AEAD Encryption Schemes

To boost efficiency when encryption and message authentication are both required, Authenticated Encryption with Associated Data (AEAD) schemes combine these two steps. By jointly defining the encryption procedures and authentication tag computation, authenticated encryption can slightly boost overall efficiency. Additionally, AEAD supports the authentication of additional non-encrypted data, which can, for example, be used to protect header fields that must be available in plaintext to interpret the ciphertext, *e.g.*, sequence numbers. Here, we take a look at Galois/Counter Mode (GCM), as one of the most widely-adopted AEAD encryption schemes.

GCM combines encryption in counter mode with a Carter-Wegman MAC. Here, the nonce n is the Initialization Vector (IV) which is incremented before being further used as counter for encrypting individual blocks, and the pseudorandom permutation F is the instantiated block cipher, usually AES. The universal hash function H then processes the not-encrypted Associated Data (AD) and the ciphertext blocks iteratively. New blocks are first multiplied by a key-dependent constant in the Galois field $GF(2^{128})$ before being XORed with the next block. The final product is XORed with the concatenated lengths of the AD and ciphertexts, multiplied one last time and finally XORed with the blinding tag to compute the authentication tag.

The important takeaway is that while AEAD schemes provide confidentiality and message authentication, message authentication is still achieved with a standalone authentication tag appended to the ciphertext. Hence, AEAD schemes also suffer from the overhead of authentication tag lengths and cryptographic processing that may limit their operability in the constrained IIoT.

2.5 MAC Aggregation

To mitigate the overhead of authentication tags that nowadays also affects the IIoT, *MAC Aggregation* has been proposed as a mitigation strategy [115]. As all the MAC schemes presented here are deterministic, *i.e.*, their verification is done by recomputing the authentication tag and comparing it to the received tag, their tags can be aggregated. If we have two tags $t_1 = Sig_{k_1}(m_1)$ and $t_2 = Sig_{k_2}(m_2)$ of the same length, we can compute an aggregated tag t by XORing t_1 and t_2, *i.e.*, $t = t_1 \oplus t_2$. The resulting aggregated tag t can only be verified by an entity knowing both messages and both keys, and a change in any of the two messages leads to a failed verification. Moreover, MAC aggregation is commutative and associative, such that aggregated tags can be aggregated with other (aggregated) tags, and the order of aggregation is not important for verification. The verification algorithms then recomputes all non-aggregated tags and recomputes the aggregated tag to verify all input messages' authenticity and integrity:

$$Vrfy(m^*_{i-n}, \ldots, m^*_i, t^*) == \begin{cases} \texttt{accept} & \text{if } Sig_{k_1}(m^*_{i-n}) \oplus \cdots \oplus Sig_{k_n}(m^*_i) = t^* \\ \texttt{reject} & \text{otherwise} \end{cases}$$

For ideal MAC schemes with independent tags, the achieved security of an aggregated tag is defined by the tag's length. This security can be easily proven by contradiction.

Proof. Consider an adversary that attempts to forge the n-bit tag t_1 for message m_1 of an ideal MAC scheme with an unknown secret key k_1. The adversary generates a tag t_2 for a message m_2 with an independent key k_2. Assume that the adversary can forge a valid aggregated tag t for the message m_1 with key k_1 and message m_2 with key k_2 with a higher probability than if they guessed a tag. The adversary could thus forge a tag t_1 with the same probability by simply computing $t_1 = t \oplus t_2$. Thus, the MAC scheme would not be ideal.□

Consequently, an n-bit aggregated tag of independent tags by ideal MAC schemes achieves a security level of n bits.

For practical applications of MAC aggregation, we sometimes want to use the same key to generate multiple aggregated tags. For example, in the XORMAC [31] scheme, one key is used to compute tags for different chunks of a message individually to parallelize expensive cryptographic operations. These tags are then aggregated to compute the final authentication tag. Similarly, aggregating the tags of a stream of short messages, generated with one key, can help to conserve bandwidth [115].

To ensure the security of the aggregated tag if one key is used, the independence of the individual tags must be ensured through other means. For example, a sufficient requirement would be that the chosen MAC scheme is *pseudorandom*, *e.g.*, HMAC [83] and that the input includes a nonce for replay protection to prevent mix-and-match attacks within one set of jointly authenticated messages [71].

3

“ Size matters not. Look at me. Judge me by my size, do you? ”

Yoda, *Star Wars: The Empire Strikes Back*, 1980

Reducing the Bandwidth Overhead of Message Authentication

One common limitation in resource-constrained environments is the available bandwidth (*cf.* Section 1.2.1). Consequently, optimizing the utilization of limited transmission resources remains an ongoing research challenge for academia and industry [218]. In this context, different streams of research explore techniques like compact protocol design [201, 202], data compression [199], data aggregation [195], or even in-network processing [74] to reduce bandwidth consumption. Furthermore, complementary approaches such as Hybrid Automatic Repeat Request (HARQ) [11] or relaying [106,221] aim at enhancing reliability, thereby decreasing the need for retransmissions over lossy channels and thus improving overall bandwidth efficiency.

Message authentication does, however, add major overhead to the bandwidth utilization as a 16-byte tag is required to achieve desirable 128 bit security levels. In this chapter, we therefore tackle our first research question: How can the bandwidth overhead of message authentication be optimized?

A recently proposed technique to reduce this overhead is progressive authentication, a novel message authentication paradigm promising to reduce authentication tag sizes by sending shorter tags. These short tags provide initially reduced security upon reception, which is improved eventually upon the reception of subsequent messages. We show that, contrary to previous claims, existing Progressive MAC (ProMAC) schemes are highly susceptible to packet loss. Thus, they cannot be deployed in many proposed constrained scenarios. In this thesis, we therefore introduce Randomized and Resilient Dependency Distributions (R2D2) that provably equip ProMAC schemes with optimal resilience to packet loss.

The fundamental idea behind ProMACs, *i.e.*, saving bandwidth for authentication tag by protecting multiple messages together, was initially proposed by Katz *et al.* [115] in 2008 for MAC aggregation, where neither immediate security demands nor lossy

channels were considered. In its simplest form, Message Authentication Code (MAC) aggregation computes a joint authentication tag over n consecutive messages and appends this tag to the last message [115]. Over the last two decades, however, communication of constrained devices in all domains has shifted significantly towards wireless transmissions. Arguably, MAC aggregation has now become even more critical in the lossy settings than for reliable channels, also because these networks expose high bandwidth constraints due to the high number of nodes sharing the same transmission medium. Due to the promising performance of ProMACs schemes, we investigate their potential for MAC aggregation in lossy channels, first simulatively, and later also by integrating and deploying MAC aggregation for DTLS 1.3.

3.1 Progressive Authentication

A major challenge of secure communication in resource-constrained scenarios stems from the overhead of integrity protection: Even the tiniest message requires a tag of several bytes (*e.g.*, 16 bytes for 128-bit security), thus significantly increasing message sizes. In this section, we first introduce ProMACs as a mechanism to reduce this overhead, before looking at its susceptibility to packet loss.

3.1.1 Core Idea and Benefits of ProMACs

Traditional MACs (*e.g.*, HMAC [29]) occupy large parts of the total payload for short messages. To partly mitigate this issue, tags can be truncated at the cost of reduced security, *i.e.*, making it easier for an adversary to guess a correct tag [194]. To provide short tags with strong security guarantees, the core idea behind ProMACs is to partially offload integrity protection into the future. Therefore, each message is initially protected with reduced security, similar to truncated MACs, but subsequent messages promptly increase this protection to an adequate level. Since tags protect multiple messages at once, ProMACs realize short tags while enabling passive resynchronization if packets get lost. This passive resynchronizability is in stark contrast to previous proposals, such as aggregated MACs [115], which jointly authenticate multiple messages with a single tag, and stateful MACs [20], which continuously reinforce the integrity of all previous messages but cannot cope with packet loss.

To cope with the low latency requirements in the Industrial IoT (IIoT), ProMACs rely on optimistic processing, *i.e.*, continuing under the assumption that all traffic is benign, and (partly) defers authenticity verification into the future. In the unlikely event of a retrospective attack detection, the system recovers from already processed malicious messages [51,160,229–232,268]. Optimistic security is especially attractive in isolated networks where attacks are relatively rare and the potential damage in a short time frame is comparable to that of less advanced attacks (*e.g.*, Denial of Service (DoS) attacks) [51,160,229–232,268]. correspondingly, optimistic processing is considered for various (lossy) scenarios such as vehicular communication [20,139,140,180,216], (industrial) IoT [20,139,140], drone control [20,24], and internal communication within hardware components (*e.g.*, Intel SGX or System-on-a-Chips (SoCs)) [20,94,164].

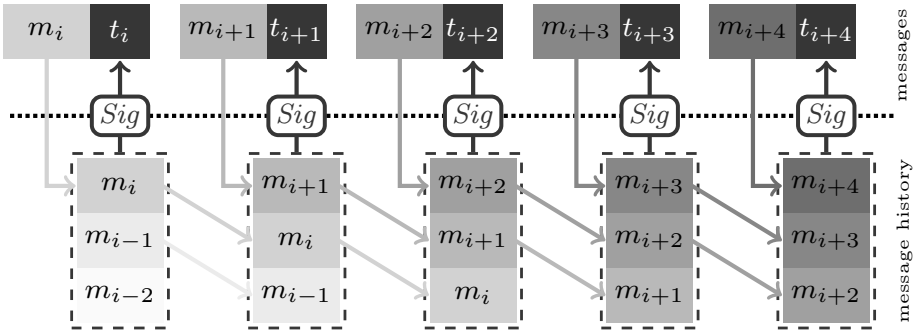


Figure 3.1 ProMACs store a history of recent messages, which is used to derive tags, effectively aggregating integrity protection of multiple messages to reduce tag sizes.

3.1.2 A Motivating Example for ProMACs

To illustrate how ProMACs' benefits manifest themselves in practice, we consider a comprehensive example from an Industrial Control System (ICS). In particular, we envision a closed-loop motion controller that reacts to continuously updated sensor readings [14]. Especially for moving systems, wireless, and thus unreliable, communication is used to avoid error-prone and expensive cable management [14]. While such a controller is resilient to the immediate impact of individual faulty sensor readings, one or multiple maliciously crafted messages can, over time, bring the system into an equipment-damaging or even life-threatening state. Meanwhile, communication channels between sensors and controllers are often constrained due to *e.g.*, a high number of network participants.

While bandwidth constraints prevent the traditional protection of each message with a 16 byte long tag, using a truncated (*i.e.*, less secure) tag hampers the reliable detection of manipulations. Therefore, an attacker could manipulate messages with long-term impact, *e.g.*, a scaling factor for speed adjustments, and thus bring the system into a critical state. While aggregated MAC schemes could eventually ensure integrity with high confidence, they would give an attacker the opportunity to manipulate multiple messages before any authenticity is verified. ProMACs mitigate this weakness by providing, albeit reduced, immediate security. ProMACs thus protect against the immediate impact of manipulated messages, while also protecting against manipulations with long-term impact.

3.1.3 Formal Definition of ProMACs

ProMACs extend traditional MACs (*cf.* Section 2.1) by additionally giving recent historical messages as input to $Sig_k(m_{i-n}, \ldots, m_i)$ and $Vrfy_k(m_{i-n}, \ldots, m_i, t)$. As shown in Figure 3.1 for a history of $n = 3$ messages, the generation of tags is then based on *multiple* messages. The tag t_{i+2}, for instance, is computed from messages m_{i+2}, m_{i+1}, and m_i. Likewise, the integrity of message m_i is protected by tags t_i, t_{i+1}, and t_{i+2}. Thus, ProMACs protect each message with multiple tags, such that each tag is only responsible for providing a fraction of the overall security. Since each tag aggregates partial integrity protection for multiple messages, progressive

integrity protection results in shorter tags. Meanwhile, a valid first tag (to the degree it can be verified) is considered sufficient to optimistically process a message, while a recovery mechanism is triggered if an attack is detected within subsequent tags. In this context, the dependencies $\mathcal{D}, \{0\} \subseteq \mathcal{D} \subset \mathbb{N}_0$, describe how the reception of one message influences the authenticity of surrounding messages. We say that a ProMAC instance has the dependencies $\mathcal{D}$, if the generation and verification of tag t_i require knowledge of $\{m_{i-d} \mid d \in \mathcal{D}\}$. Consequently, a message m_i blends into all tags $\{t_{i+d} \mid d \in \mathcal{D}\}$ and a tag t_i protects the integrity of all messages $\{m_{i-d} \mid d \in \mathcal{D}\}$.

3.1.4 Existing ProMAC Schemes

In practice, the theoretical notion of ProMACs has been realized by at least three schemes: *Whips* [20], CuMAC [139,140], and *Mini-MAC* [216]. Our discussion of these approaches focuses on their selection of identical dependencies $\mathcal{D}$, as those describe how the failure to receive one message influences the verifiability of neighboring tags.

Whips. Whips [20] was proposed alongside the formal introduction of ProMACs. It provides a fixed security level of 128 bit with a constant memory overhead per message stream. To this end, Whips tracks the message history via an internal state s, used to derive tags t, that is composed of a counter c (for replay protection) and a fixed number n of substates $\tilde{s}$. The number of substates is inversely proportional to the targeted tag lengths, such that if smaller tags are used, a message's integrity is protected by more tags. Each substate $\tilde{s}_i$ corresponds to exactly one message and is computed as $\tilde{s}_i = \mathit{trunc}(\mathtt{HMAC}_k(m_i))$. The size of $\tilde{s}$ depends on the targeted security level (*e.g.*, $\tilde{s}$ has to be at least 32 byte long for 128-bit security [20]). To generate a new tag t_i for a message m_i, the state s_i is first updated by (i) incrementing the counter c, (ii) appending the substate $\tilde{s}_i$ to s, and (iii) removing the substate $\tilde{s}_{i-n}$ from s. The tag t_i for message m_i is then computed as $\mathit{trunc}(\mathtt{HMAC}_k(s_i))$. Since a tag thus depends on the last $n-1$ messages, Whips relies on sliding window-based dependencies, *i.e.*, $\mathcal{D} = \{0, \ldots, n-1\}$.

CuMAC. Simultaneous to the formalization of ProMACs [20], CuMAC [140] proposed a similar concept with cumulative message authentication codes. In CuMAC, first, a traditional MAC σ is computed from a counter c and a message m. Then, σ is split into n fragments of equal length, *i.e.*, $\sigma = \sigma^0 || \ldots || \sigma^{n-1}$. Finally, the tag t_i for message m_i is computed by aggregating fragments of the authentication tags σ for the n past messages using XOR (one distinct fragment per message), *i.e.*, $t_i = \sigma_i^0 \oplus \sigma_{i-1}^1 \oplus \cdots \oplus \sigma_{i-n-1}^{n-1}$. Thus, CuMAC also relies on a sliding window of the n most recent messages ($\mathcal{D} = \{0, \ldots, n-1\}$). CuMAC can be further improved with CuMAC/S [139] in scenarios with predictable traffic patterns by predicting future messages and pre-authenticating these messages to achieve immediate full authentication upon message reception. However, this does not change the dependencies $\mathcal{D}$.

Mini-MAC. Mini-MAC [216] re-authenticates CAN bus messages within subsequent messages to address the problem of insufficient payload size. Although originally not designed as generally applicable, retrospectively, Mini-MAC can be interpreted as ProMAC scheme if we ignore optional extensions. Mini-MAC derives a tag t_i for

message m_i from a sliding window of the n most recent messages ($\mathcal{D} = \{0, \ldots, n-1\}$) and a counter c for replay protection: $t_i = trunc(\texttt{HMAC}_k(c \parallel m_{i-(n-1)} \parallel dots \parallel m_i))$. The size of the sliding window n is not fixed. A larger n results in higher eventual security by authenticating messages more often, but also requires more computations and increases the impact of transmission failures. Additionally, t_i is truncated to the space remaining in the given packet. Consequently, Mini-MAC provides integrity protection in a best-effort manner.

3.1.5 Security Considerations for ProMACs

Security of ProMACs so far centered around an attacker with the same goal and means as for traditional MACs, *i.e.*, attacking individual packets by guessing keys or forging tags [20, 139, 140, 216]. In this setting, ProMACs provide at least the same security as traditional MACs: For the latest message, the security of ProMACs is allegedly identical to traditional (truncated) MACs and becomes stronger with subsequent packets [20]. However, these security considerations ignore the impact of *lost or dropped* packets, which influence the verifiability of neighboring tags. Hence, we extend the ProMAC threat model and show that this leads to novel attacks that severely limit the applicability of current ProMACs on lossy channels.

3.1.5.1 Extended Threat Model

To accommodate for ProMACs spreading authenticity over multiple packets, some of which may be lost due to a lossy (*e.g.*, wireless) channel, we extend the original threat model of ProMACs [20] in two ways. Firstly, we extend the attackers' capability beyond simply observing and querying message-tag pairs by giving them the additional capability of inducing and reacting to transmission failures. Secondly, we alter the attackers' goals to include not only the forging of valid tags for a previously unseen message but also the disruption of the communication channel by *e.g.*, amplifying a DoS attack by abusing the characteristics of ProMACs.

3.1.5.2 Sandwich Attack Against Current ProMACs

So far, ProMACs have not considered the effects of transmission failures, either caused by a lossy channel or active interference, in their (formal) security proofs [20, 139]. The *sandwich attack* against ProMACs presented in this thesis leverages exactly this attack vector: If two transmission failures are less than the tracked message history n apart, all messages "sandwiched" between these failures remain unauthenticable. Selective jamming to induce these failures is, however, hardly distinguishable from random packet loss, such that these attacks are hard to detect, as only a small number of dropped packets can have detrimental consequences.

Figure 3.2 explains the root cause of this attack using an example with a message history of length $n = 3$ (chosen short for illustration). We assume that message m_i is not received, either due to a transmission failure or jamming. Then, because

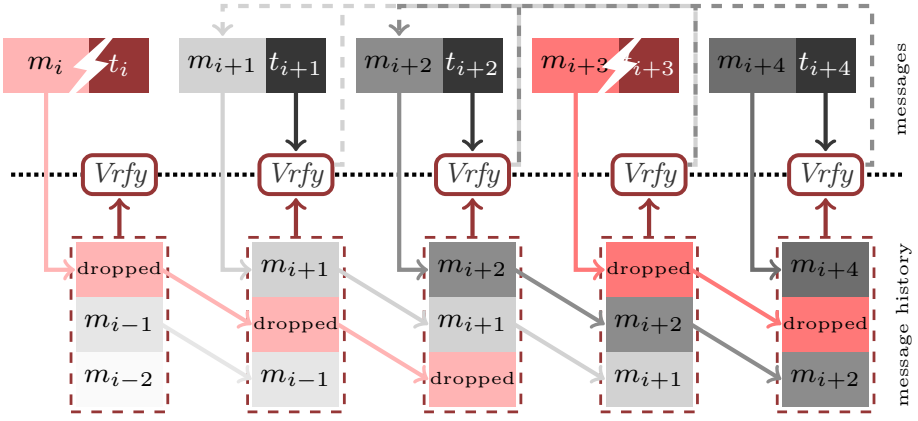


Figure 3.2 Sliding window-based dependencies, as used by all current state-of-the-art ProMACs schemes, allow an attacker to remove integrity protection from multiple messages by sandwiching them between dropped packets.

of the dependencies $\mathcal{D} = \{k \mid 0 \leq k < n\}$ of current ProMACs, all future tags t_{i+k} $(k < n)$ cannot be verified. In itself, this is not a serious problem, as eventually, messages m_{i+k} $(k < n)$ will still receive (reduced) integrity protection through the tags t_{i+k+j} $(n - k + i < j < n)$. Therefore, in the primarily envisioned scenarios for ProMACs, m_{i+1} and m_{i+2} would be processed optimistically under the assumption that messages stemming from an attacker would be detected before any real damage could occur. However, if another message m_j $(i + 1 < j \leq n)$ is also not received (by chance or triggered by interference), then all tags $\{t_i, t_{i+1}, \ldots, t_{j+n-1}, t_{j+n}\}$ cannot be verified. Consequently, all messages m_l $(i < l < j)$, sent in between m_i and m_j, cannot be authenticated, as their integrity protection relies on the tags $\{t_{l+k} \mid k \in \mathcal{D}\} \subset \{t_i, t_{i+1}, \ldots, t_{j+n-1}, t_{j+n}\}$. Already for our selected short message history, the authenticity of m_{i+1} and m_{i+2} cannot be verified despite being received correctly.

Overall, ProMACs' prospects to bandwidth-efficiently protect lossy communication are highly desirable. However, their susceptibility to network-level disturbances limits deployability. Consequently, those scenarios that benefit most from ProMACs' bandwidth savings are the most vulnerable to the presented sandwich attack.

3.1.5.3 Implications of the Sandwich Attack

Current ProMAC schemes suffer from a common vulnerability inherent to their design: Since integrity protection is distributed among *consecutive* messages, the (forced) loss of two messages that are less than the message history apart already disables integrity protection for all messages in between. Thus, with minimal and unsuspicious interference, an attacker can covertly remove authenticity from multiple ProMAC-protected messages. Depending on how ProMACs are deployed, this vulnerability leads to effective denial-of-service attacks or even false data injections.

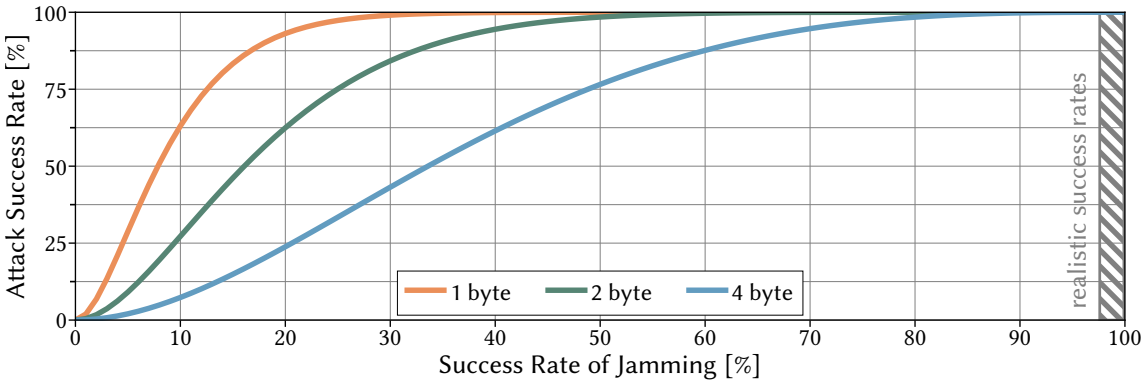


Figure 3.3 How well a selective jammer can execute the sandwich attack depends on the effectiveness of jamming. In realistic scenarios, the attack can be launched reliably even from an imperfect jammer.

3.1.5.3.1 Reverting All Suspicious Traffic

ProMACs promise to enable the optimistic processing of messages upon reception based on reduced initial security. In the unlikely event of a retroactive detection that a message could not be authenticated, the effects of a potentially malicious message have to be reverted. If ProMACs operate on a high-reliability link with hardly any packet loss, this mode of operation is reasonable. However, in the presence of a jamming attack or a less reliable channel, current ProMACs schemes cannot ensure a stable operation due to frequent rollbacks. Consider a *selective jammer* that listens to the communication channel to identify "interesting" packets, *i.e.*, those that need to be suppressed to launch a sandwich attack, and then deliberately distorts these packets [19, 97, 257]. To implement this jammer, an attacker must control a device that (i) is in range of the targeted channel and (ii) can jam ongoing communication.

In Figure 3.3, we study how jamming capabilities translate to success rates for the sandwich attack. A successful attack means that a considered packet could not be authenticated, which in this deployment scenario forces the entire system to roll back to the point at which the targeted message has been optimistically processed. To this end, we assume that an attacker attempts to make exactly one message unauthenticable by jamming the neighboring packets influencing this message's integrity. We consider three tag sizes (1, 2, and 4 bytes) and analytically compute the attack success rate for varying likelihoods of successful jamming. Our results show that even an imperfect jammer can execute the sandwich attack reliably. Furthermore, smaller tags are more susceptible to selective jammers (for identical security levels), as their larger sliding windows (which are inversely proportional to tag sizes) give an attacker more opportunities to jam packets.

To put our results into perspective, we highlight (● in Figure 3.3) the practical likelihood of successful selective jamming [19, 257]. For IEEE 802.15.4, used in common wireless protocol stacks for constrained devices, selective jamming has proven effective in covertly dropping packets with success rates between 97.6% and 99.9% [257]. Similar numbers have been reported for LoRaWAN, used for energy-efficient long-range IoT communication, where selective jamming using commodity hardware shows success rates between 98.7% and 99.9% [19]. These success rates

Channel	Model Parameters (%) p	r	$\texttt{err}_{\text{good}}$	$\texttt{err}_{\text{bad}}$	resulting avg. PER
low-error	0.5	75.9	1.1	62.4	1.50%
high-error	3.2	83	6.8	78.8	9.47%

Table 3.1 Our parametrization of the Gilbert-Elliot models of a realistic low- and high-error link based on recent longitudinal measurements of industrial networks [99].

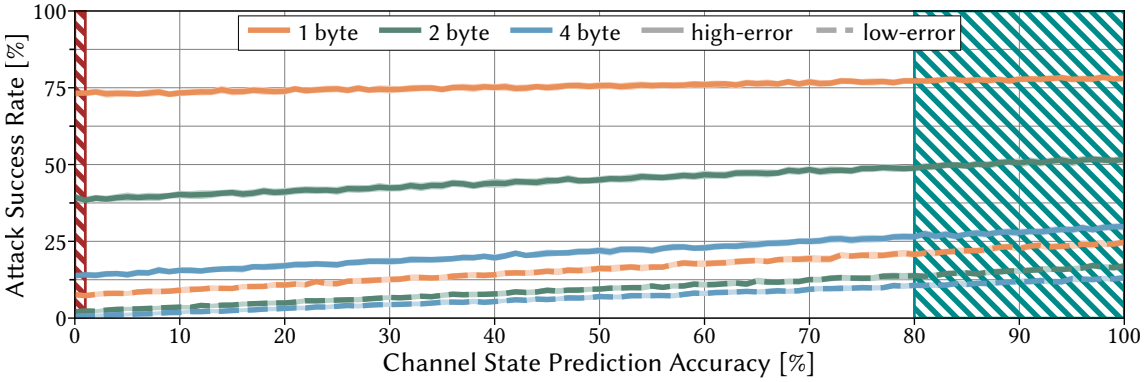


Figure 3.4 Even attackers without jamming capabilities can execute the sandwich attack, especially if they can predict the state of the communication channel well enough.

can be further improved if transmission times can be predicted, *e.g.*, when TDMA is used on the medium access layer [97]. Considering these numbers, our analysis shows that active jammers can execute the sandwich attack with success rates above 99.9% in realistic settings. Consequently, a jammer can cripple a ProMAC-protected communication link with selective, and thus stealthy, interference.

Furthermore, current ProMACs cannot be exposed to harsh environments, such as ICS with realistic error rates of 1 to 10 % [236, 255], without restrictions. To validate this claim, we simulate two wireless communication channels ("low-error" and "high-error") based on the Gilbert-Elliot (G-E) model, commonly used to simulate wireless channels based on a Markov chain with two states [60]. In the G-E model, the two states are used to encode a "good" and "bad" channel state, with corresponding packet drop probabilities $\texttt{err}_{\text{good}}$ and $\texttt{err}_{\text{bad}}$. The parameters `p` and `r` define the probability for switching from "good" to "bad" and vice versa. We parameterize the G-E models as summarized in Table 3.1 based on recommendations from literature [99], to represent packet error rates of approximately 1% (low-error) and 10% (high-error), which are realistic for scenarios envisioned for ProMACs [236, 255].

Using these two channel models, we first investigate the number of unauthenticatable packets for varying tag sizes (1, 2, and 4 bytes). Figure 3.4 reports on the mean attack success rate over 30 Monte Carlo simulations covering 1000 attacks with 99% confidence intervals. A successful "attack" again means that the considered packet could not be authenticated. We found that the overall channel quality, as well as the used ProMAC tag length, influence the attack's success rate. In particular, we observe (● in Figure 3.4) that between 0.6 % and 73.1 % of messages do not have any verifiable integrity protection, depending on the overall channel quality and tag

length. These results indicate the number of overall unauthenticable packets for current ProMAC schemes over lossy channels.

3.1.5.3.2 Reacting Only to Explicitly Detected Attacks

Previous publications on ProMACs [20, 139, 140, 216] imply a second deployment scenario, where rollbacks only take place after a manipulation is *explicitly* detected. However, in this scenario, an attacker can inject malicious traffic into a data stream that is not detected as such by jamming the neighboring packets with a high success rate (*cf.* Figure 3.3). Alternatively, a less powerful attacker can abuse the naturally occurring unauthenticatable packets. If attackers are even able to predict a bad channel state, they can increase the chances of their data injection not being detected. To illustrate this behavior, we increase the channel state prediction accuracies (specifying the likelihood of the channel being in the "bad" state) for the attacker on the x-axis in Figure 3.4. After predicting a bad channel, the attacker waits for one additional transmission before injecting a forged message. The attack is successful if the two windows, starting and ending with the forged message, each contain at least one transmission failure.

Such prediction accuracies can be upwards of 80 % if the attacker is in the vicinity of the receiver (<1 m apart), as shown by practical measurements [271]. Assuming an accuracy of 80 %, an attacker can successfully launch a sandwich attack in 10.4 % to 75.4 % of attempts, depending on the link quality and tag lengths (● in Figure 3.4).

3.2 Designing ProMACs Resilient to Packet Loss

Overall, the prospects of ProMACs are extremely desirable, but even a moderate Packet Error Rate (PER) on the transmission medium leads to a significant risk of false data injections or crippled communication channels. Meanwhile, smaller tags with larger sliding windows favor the sandwich attack exactly for those scenarios that benefit the most from ProMACs. This weakness of current ProMAC schemes stems from an inherent design choice: By spreading a message's integrity protection over *consecutive messages*, these schemes become susceptible to packet loss, where the (malicious) interference on a few packets invalidates the authenticity for multiple messages. To eliminate this attack vector, it is necessary to *interleave message dependencies* to better distribute the effects of dropped packets to prevent individually lost packets from reducing the protection of targeted messages to insecure levels.

3.2.1 R2D2: ProMAC Schemes with Optimal Resilience

While the general idea of interleaving message dependencies seems promising, it requires finding the right trade-off between delay for full integrity protection and resilience to dropped packets. To achieve this goal, we propose Randomized and Resilient Dependency Distribution (R2D2) as a foundation for ProMACs that are

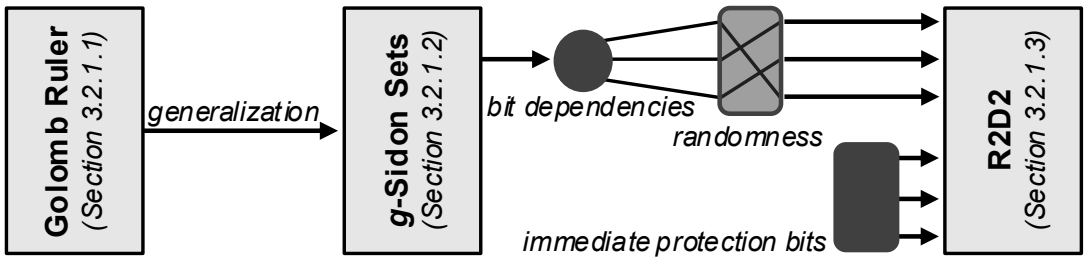


Figure 3.5 R2D2 combines different theoretical building blocks to realize randomized and resilient dependency distribution and to thwart sandwich attacks against ProMACs.

resilient to network-level interference. As shown in Figure 3.5, R2D2 is based on *optimally* interleaved dependencies (Section 3.2.1.1) and a generalization of this concept to achieve *parameterized security guarantees* (Section 3.2.1.2). R2D2 enhances this foundation through *bit dependencies*, *randomization*, and *immediate protection bits* (Section 3.2.1.3).

3.2.1.1 Golomb Ruler-based Dependencies

Existing ProMAC schemes rely on a sliding window for their dependency distribution, where each tag's computation requires knowledge of the last n consecutive packets, *i.e.*, $\mathcal{D} = \{0, \ldots, n\}$. However, exactly this property is abused by the sandwich attack to void the integrity protection of targeted messages. To mitigate this weakness, ProMACs have to interleave dependencies such that the effects of dropped packets are cushioned by a large set of tags. To achieve this goal, we propose to use *Golomb Rulers* [23,225], which are used, *e.g.*, in radio astronomy to determine optimal antenna placements [235], to minimize overlap between tags of different messages. Intuitively, a Golomb Ruler is a set of integer marks on a discrete ruler, placed such that the distance between any pair of marks is unique. Formally, a set S ($\{0\} \subseteq S \subset \mathbb{N}_0$) is a Golomb Ruler iff $\forall s_1, s_2, s_3, s_4 \in S$ with $s_1 \neq s_2$ and $s_3 \neq s_4$ it holds that $s_1 - s_2 = s_3 - s_4 \iff s_1 = s_3$ and $s_2 = s_4$. The length of a Golomb Ruler is defined as the value of its largest element.

Golomb Rulers provide the theoretical foundation to realize message dependencies that minimize the overlap between tags of neighboring messages. As the distance between two tags protecting a specific message is unique, Golomb Rulers guarantee that a message's security level is reduced by at most the integrity protection provided by one tag for any dropped message:

Proposition 1. By dropping a message m of a ProMAC-protected stream, any other message's integrity protection is reduced by at most the security provided by one tag iff the dependency $\mathcal{D}$ is a Golomb Ruler.

Proof. Without loss of generality, we investigate the authenticity of the message m_i. The integrity of m_i is protected by tags $\{t_{i+d} \,|\, d \in \mathcal{D}\} = \mathcal{P}$. We prove our claim by contradiction. Therefore, we assume that there exists m_j $(j \neq i)$, such

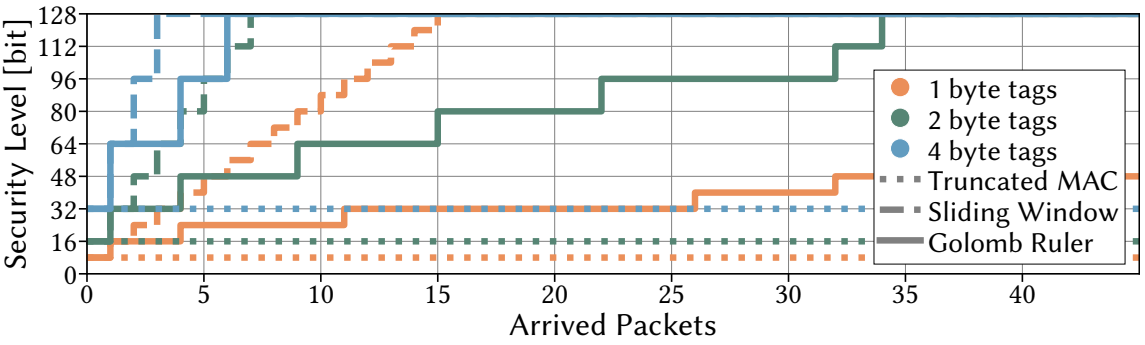
that at least two tags from $\mathcal{P}$ become unverifiable if m_j is dropped. These two tags have the form $t_{i+d} \in \mathcal{P}$ and $t_{i+d'} \in \mathcal{P}$ $(d \neq d')$. If both of these tags become unverifiable because m_j is dropped, m_j has to be included in the intersection $\mathcal{I} = \{m_{i+d-\delta} \mid \delta \in \mathcal{D}\} \cap \{m_{i+d'-\delta'} \mid \delta' \in \mathcal{D}\}$ of the messages that are required to compute these tags. Since we assumed that $m_j \in \mathcal{I}$, this requires the existence of $d, d', \delta, \delta' \in \mathcal{D}$, such that $d - \delta = d' - \delta'$. However, exactly when $\mathcal{D}$ is a Golomb Ruler, this only holds if both differences equal 0, *i.e.*, $d = \delta$ and $d' = \delta'$. However, if $d - \delta = d' - \delta' = 0$, then it has to hold that $\mathcal{I} = \{m_i\}$, which means that m_j $(i \neq j) \notin \mathcal{I}$. Thus, there cannot exist any message m_j that, if dropped, invalidates multiple tags authenticating m_i. □

Exemplarily, using the Golomb Ruler $\{0, 1, 4, 6\}$ of length 6 ensures that any dropped message only invalidates at most one tag protecting the integrity of any other message, while providing full security guarantee after 6 subsequent messages have been received. Using the shortest Golomb Rulers for a given number of elements, *i.e.*, an optimal Golomb Ruler, as in the previous example, thus provides the mentioned security guarantees within the shortest possible delay until full authenticity is reached.

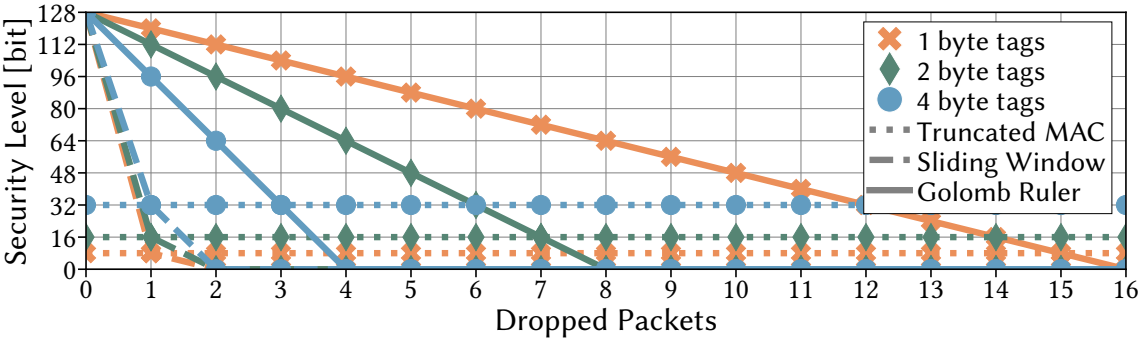
We now have to investigate what Golomb Ruler-based dependencies mean in general for the security of ProMACs, based on two core metrics: (i) the *delay* to reach full protection, and (ii) the *resilience* to targeted packet dropping. Here, delay is expressed as transmitted packets, as actual time depends on the communication pattern of the underlying application. Resilience is expressed by the minimum obtainable bit security of any message for a given number of lost packets. To express resilience in terms of bit security, we assume that a l-bit tag provides exactly l bits of security, *i.e.*, 128-bit security is realized by appending a 16-byte tag to a message.

In Figure 3.6, we show the delay and resilience of Golomb Rulers-based ProMACs by comparing them to truncated MACs and sliding windows-based ProMACs for tags that are 1, 2, and 4 byte long. Figure 3.6a shows how the protection develops over time in absence of an attacker. We observe a fast increase in security for sliding window-based ProMACs and no variation in the provided security over time for truncated MACs. Using optimal Golomb Rulers as dependencies, the delay until full security is reached is acceptable for tag sizes of 4 and 2 byte, whereas 1-byte tags only reach full security after receiving 177 additional messages.

In contrast, Figure 3.6b shows the resilience of different schemes against network-level attacks. Here, the susceptibility to the sandwich attack of sliding window-based ProMACs can be seen again, as the provided protection of a message can be rendered void with just 2 dropped packets. Truncated MACs are, as expected, not susceptible to network-level attacks. When it comes to the resilience of Golomb Ruler-based ProMACs, we see an inverted behavior as in Figure 3.6a. 1 and 2 byte long tags provide significant resilience to network-level attacks, remaining well over the security level of truncated MAC, even if a high fraction of the relevant packets are dropped. However, longer tags remain susceptible to variants of the sandwich attack, *i.e.*, the integrity of messages protected by 4 byte long ProMACs can be attacked by dropping 4 targeted messages.



(a) Golomb Ruler-based dependencies require the reception of more messages to achieve full 128-bit security than sliding window approaches, and thus increase the delay of integrity protection in a message stream, particularly for small tags (≤2 byte).



(b) Dropped packets have an obvious impact on the progressive nature of ProMACs. However, the resiliency to dropped messages is significantly increased by Golomb Ruler-based ProMACs compared to current state-of-the-art ProMACs.

Figure 3.6 Golomb Ruler-based ProMACs (in comparison to sliding window-based ProMACs and truncated MACs) protect against network-level attacks, albeit with a stiff tag length-dependent trade-off between speed and security.

Golomb Ruler-based dependencies provably minimize the delay until full security is achieved while ensuring that a dropped packet impacts at most one tag protecting any other message. However, the resulting inflexible trade-off between achievable delay and resilience might not match the requirements of specific use cases. We thus discuss how R2D2 addresses this challenge via generalized Golomb Rulers.

3.2.1.2 Tag Length-independent Security Levels

Message dependencies with minimal overlap based on Golomb Rulers directly couple the security loss from a dropped packet to the length (and thus bit security) of individual tags (*cf.* Section 3.2.1.1). Thus, while providing optimal dependencies w.r.t. the number of unverifiable tags in any given message, these dependencies may result in unacceptable verification delays for certain scenarios. To resolve this stiff trade-off, we propose *tag length-independent security levels* for message dependencies that enable a parametrization of the maximum security loss per dropped packet.

This parametrization enables to definition of tags of different sizes that each provide similar resilience to network-level attacks. The core idea to achieve tag length-

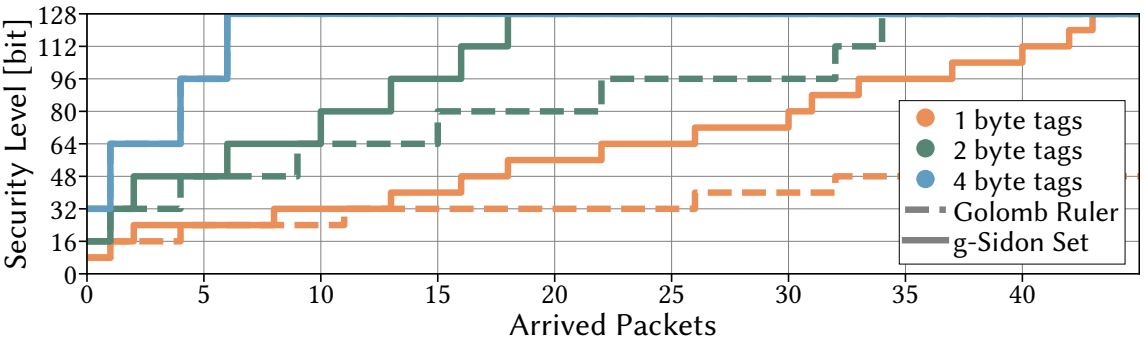
independent security levels in R2D2 is to give control over how many tags, protecting the integrity of a single message, become at most unverifiable through a dropped packet. To realize this idea, we use *g-Sidon Sets* [225], a generalization of Golomb Rulers. Intuitively, a g-Sidon Set is a set of integer marks on a discrete ruler, which are placed such that the distance between any pair of marks occurs at most g times. Formally, a set $S \subset \mathbb{N}_0$ is a g-Sidon Set iff any pairwise difference between elements occurs at most g times, *i.e.*, S is a g-Sidon Sets iff there exist at most g distinct pairs ($s_0 \in S$, $s_1 \in S$) with $s_0 < s_1$ such that $s_0 - s_1 = k$, for all $k \in \mathbb{Z}^*$. Using g-Sidon Sets as dependencies $\mathcal{D}$ guarantees that any message's security level is reduced by at most the integrity protection provided by g tags for any dropped message:

Proposition 2. Using g-Sidon Sets as ProMAC dependencies $\mathcal{D}$ guarantees that any message's security level is reduced by at most the integrity protection provided by g tags for any dropped message.

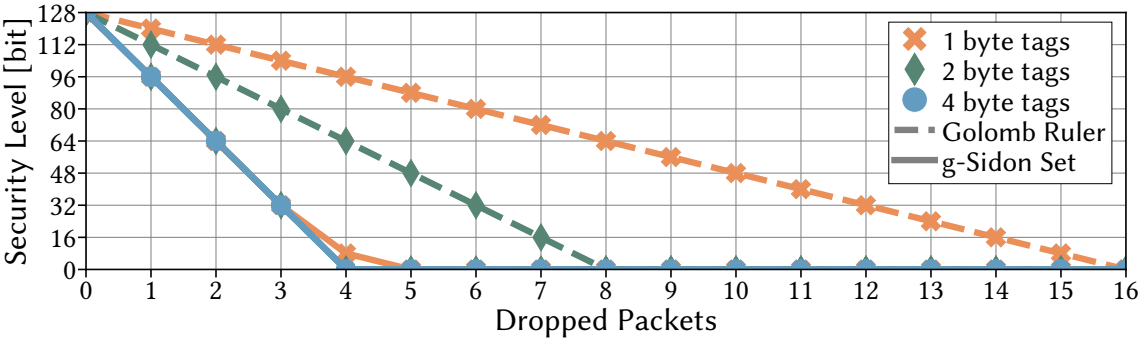
Proof. Without loss of generality, we investigate the authenticity of the message m_i. The integrity of m_i is protected by tags $\{t_{i+d} \mid d \in \mathcal{D}\} = \mathcal{P}$. We prove Proposition 2 by contradiction. Therefore, we assume that there exists m_j $(j \neq i)$, such that at least $g + 1$ distinct tags from $\mathcal{P}$ become unverifiable if m_j is dropped. These $g + 1$ tags have the form $t_{i+d} \in \mathcal{P}$, with a distinct d for each tag. If all of these $g + 1$ tags become unverifiable because m_j is dropped, m_j has to be included in the intersection $\mathcal{I} = \{m_{i+d-\delta} \mid \delta \in \mathcal{D}\} \cap \{m_{i+d'-\delta'} \mid \delta' \in \mathcal{D}\}$ of the messages that are required to compute these tags. Since we assumed that $m_j \in \mathcal{I}$, this requires the existence of $g + 1$ distinct δ-δ' pairs, such that $d - \delta = d' - \delta'$, with $d, d', \delta, \delta' \in \mathcal{D}$. However, exactly when $\mathcal{D}$ is a g-Sidon Set, there exist by definition at most g distinct δ-δ' pairs (*cf.* Section 3.2.1.2). Thus, there cannot exist a message m_j that, if dropped, invalidates more than g tags that authenticate m_i. □

Thus, Golomb Rulers are 1-Sidon Sets, as any difference between elements in a Golomb Ruler is unique. Overall, these provable and parameterized levels of protection against network-level attacks are possible iff g-Sidon Sets are used as dependencies, while their optimality guarantees that full security is achieved in the fastest possible way.

For ProMACs, g-Sidon Sets thus promise to efficiently parameterize the maximum security loss for dropped packets in terms of bit security to decouple this property from the tag length and gain control over verification delays. To verify this claim, we compare verification delays and resilience to network-level attacks of Golomb Ruler-based dependencies to those based on g-Sidon Sets in Figure 3.7. We choose g such that a dropped message induces at most a 32-bit security loss, independent of the underlying tag size (*i.e.*, $g = 1$ for 4-byte tags, $g = 2$ for 2-byte tags, and $g = 4$ for 1-byte tags). Figure 3.7a shows the improvements to verification delays based on parameterization. By allowing dependencies to overlap twice, we can nearly halve the verification delay of 2-byte tags, and the verification delay of 1-byte tags can be reduced from 177 to 43. However, this speedup also reduces the resilience to attacks, as can be seen in Figure 3.7b. Here, we show the advantage of g-Sidon Sets-based dependencies since all parameterizations lose security to a similar extent with the number of dropped packets, but this loss is bounded by the maximal security loss of



(a) Using g-Sidon Sets instead of Golomb Rulers can significantly reduce the delay until the full security level is reached.



(b) ProMACs based on g-Sidon Sets enable parameterized tag length-independent security loss resulting from dropped packets.

Figure 3.7 g-Sidon Sets, in contrast to Golomb Rulers, enable to control the security loss per dropped packet (here: 32 bits).

32 bits per dropped packet. Thus, a variation of the sandwich attack would require the targeted dropping of at least four packets to remove all authenticity.

Dependencies based on g-Sidon Sets thus achieve tag length-independent security levels and allow a flexible parameterization of the trade-off between verification delays and resilience to packet loss. We observe that full security can be provided significantly faster for smaller tags by optimal g-Sidon Sets than by Golomb Rulers-based dependencies, which is counteracted by a reduction in resilience to packet loss. Additionally, using optimal g-Sidon Sets lets an attacker know the optimal strategy to remove integrity protection from a targeted message, whereas it would be advantageous to hide this strategy. In the following, we see how R2D2 addresses these remaining weaknesses through bit dependencies, hiding of the optimal attack strategy, and immediate protection bits.

3.2.1.3 Secure Dependencies through R2D2

Both message dependencies with minimal overlap (Section 3.2.1.1) and its more flexible generalization for tag length-independent security levels (Section 3.2.1.2) realize optimal and thus *deterministic* dependencies for their respective parametrizations. Consequently, while these improved dependencies considerably increase the number

of necessary packet drops to disable integrity protection, an attack can still leverage this determinism to derive *which* packets to drop.

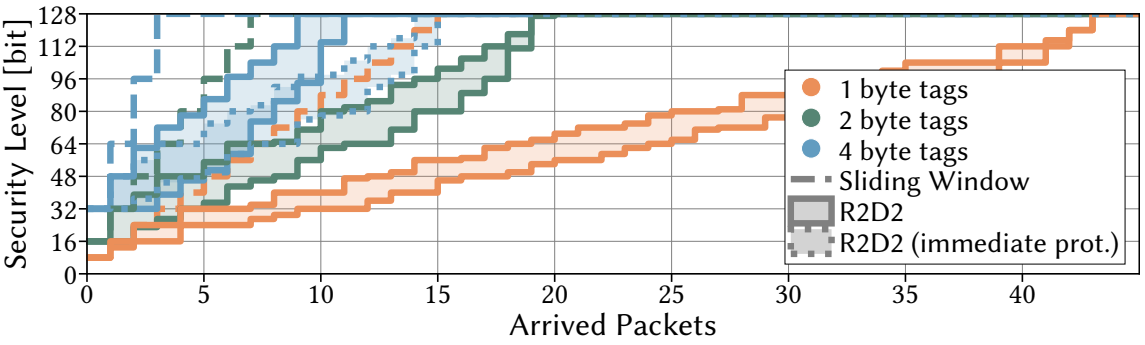
R2D2 addresses this issue by *randomizing* dependencies to hide which messages have to be dropped. Further enhancing this approach, R2D2 introduces *bit dependencies*, *i.e.*, each bit of a tag protects a different message set. Each R2D2 instance is initialized with a *pseudorandom* set of dependencies $\mathfrak{D} = \{\mathcal{D}_0, \mathcal{D}_1, \dots\}$, where the number of dependencies equals the tag length, *i.e.*, $|\mathfrak{D}| = |t|$. Each dependency $\mathcal{D}_i$ is a g-Sidon Set, where its order, *i.e.*, number of elements, depends on the tag length $|t|$ such that the total number of dependencies equals the targeted security level, *e.g.*, $\sum_{0 \leq i < |t|} |\mathcal{D}_i| = 128$. The parametrization of g follows from the tolerable security loss (*cf.* Section 3.2.1.2). The set of dependencies $\mathfrak{D}$ is pseudorandomly sampled (using a shared key between sender and receiver) from the n precomputed most optimal g-Sidon Sets.

Instead of selecting one of n potential dependencies, the number of potential distributions increases to $\binom{n}{|t|}$ through randomized bit dependencies. This increased variety enables strong resilience with a relatively short n, *e.g.*, 64, meaning that verification delay remains low and that even constrained devices can store the set of potential bit dependencies. Additionally, by using dependencies of different orders, R2D2 can achieve a specific security level, *e.g.*, 128 bits, even if the targeted tag length does not divide the security level, since the achieved security level amounts to $\sum_{0 \leq i < |t|} |\mathcal{D}_i|$ bits.

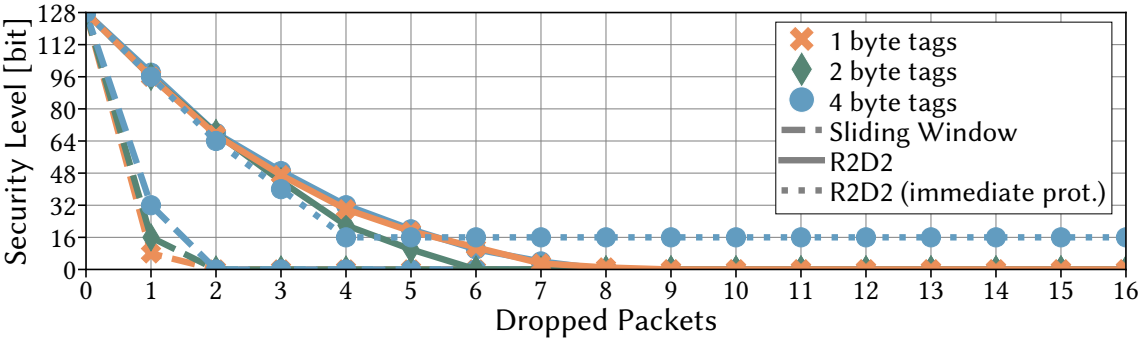
As an additional benefit, bit dependencies enable *immediate protection bits*. Those only depend on the current message ($\mathcal{D}_i = \{0\}$) and thus, like truncated MACs, are resilient to network-level interference. This ensures that, no matter how many packets are dropped by an attacker, the protection of a received message is never completely removed. To still reach the targeted bit security level when using immediate protection bits, the order of the remaining dependencies $\mathcal{D}_i \in \mathfrak{D}$ has to be increased accordingly.

The concepts of randomized bit dependencies and immediate protection bits promise to increase the resilience to network-level attackers without significant impacts on the verification delay. To verify this claim, we again compare the verification delay and resilience of R2D2 to sliding window-based dependencies, the current state-of-the-art in Figure 3.8. In addition to the 1, 2, and 4 byte tags with randomized bit dependencies, we also consider the case where half of a 4-byte tag is reserved for immediate protection bits. As before, all tag lengths are parameterized to allow a maximal security loss of 32 bits per dropped packet. All dependencies are selected from the 64 shortest ones for a given parametrization, *i.e.*, $n = 64$.

In Figure 3.8a, the verification delay of R2D2 constitutes an area between the minimum and maximum delay depending on which dependencies are randomly selected. Overall, we observe similar delays as in Figure 3.7a. Additionally, in Figure 3.8b, we show the worst-case resilience of R2D2 against a network-level attacker. Therefore, we assume that (i) the attacker knows the selected bit dependencies, and (ii) that the most vulnerable dependencies are selected for the most efficient variation of the sandwich attack. We observe that the resilience of R2D2 increases even in this worst-case scenario through the introduction of bit dependencies, while in reality, the



(a) R2D2 introduces randomness into its bit dependencies without significantly increasing the delay for full integrity protection.



(b) R2D2 protects against sandwich attacks, even if the attacker learns the randomized bit dependencies (worst-case assumption).

Figure 3.8 Even in the worst-case (*i.e.*, the attacker somehow learns the secret bit dependencies), R2D2 offers high resilience to network-level attacks without significant compromises in terms of verification delay.

attack would require to drop even more packets as he cannot be sure which packets need to be dropped to execute a sandwich attack. In practice, this protection against targeted packet drop is even higher as the selected dependencies remain secret.

Considering the introduction of immediate protection bits to 4-byte tags, we see in Figure 3.8a that this addition results in a slight increase in the delay until full protection is achieved. The reason for this additional delay is that to still target 128-bit security, higher-order dependencies have to be chosen because fewer bits are available for progressive authentication. Looking back at Figure 3.8b, we see that this additional delay creates baseline protection that is not susceptible to network-level attacks, no matter how many packets an attacker can drop.

3.2.1.4 Security Properties of R2D2

Overall, R2D2 defines parametrizable dependencies which increase the resilience of ProMACs to packet drops, either through a bad channel or malicious activities (*i.e.*, sandwich attacks and variations thereof). In addition to inheriting the security guarantees of g-Sidon Set-based dependencies, *i.e.*, parameterizable bounds for the maximal security loss per dropped packet, R2D2 additionally hides the optimal attack strategy for an attacker. A big strength of R2D2 is its flexibility: ProMACs can be

adapted to different use cases by choosing trade-offs between the maximal security loss per dropped packet, tag lengths, and acceptable delays until full authenticity can be provided. Finally, R2D2 achieves provably *optimal* delays for given security parameters, such that we can understand where ProMACs may not be applicable.

Practical Authentication Delays. As ProMACs provide only reduced initial security, messages are processed optimistically and may retrospectively be detected as malicious. Related work on optimistic security [51, 180, 229, 231, 232] puts resulting delays into perspective to understand where ProMACs are applicable. For intra-vehicular communication, Szilagyi *et al.*, *e.g.*, demonstrate that authentication delays of up to 100 messages are acceptable even for throttling control due to high sampling rates and physical inertia [229, 232]. Similarly, Nilsson *et al.* conclude that delays of up to 16 messages are easy to recover from [180]. For ICSs, Castellanos *et al.* [51] propose acceptable reaction times of 60 ms (corresponding to 100 packets) for the SWaT water treatment testbed [86], which could, however, be considered more inert than other latency-critical domains. Szilagyi *et al.* argue that ICSs may require multiple message forgeries to bring it into an unsafe state [231], which would provide time to optimistic processing.

While R2D2 can be parameterized to individual needs, 2 and 4 byte long tags (latter with 16 immediate protection bits) with a maximum security loss of 32 bit are attractive in many real-world deployments: Packets are adequately protected against network-level attacks and reach full security within acceptable delays. However, R2D2's optimal delays show that shorter tags or strong security guarantees with ProMACs, in general, can only be provided if longer delays are acceptable.

ProMACs on High Error Rate Channels. R2D2 lowers the number of ProMAC-protected messages with unverifiable authenticity due to normal packet loss. Yet, we still measure unverifiable authenticity for around 10% of messages for 1-byte tags on a channel with a packet loss of 9.1%. Thus, for adequate security, the use of R2D2's immediate security bits and slightly longer tags is crucial to realize secure ProMACs for such channels. Meanwhile, for the same 1-byte tags, we did not observe a single message that lost all authenticity over tens of millions of transmitted packets with a packet loss of 0.9%. Thus, for higher reliability channels, R2D2 makes it unlikely that a processed message has to be reverted because it could not be retroactively authenticated without malicious interference.

3.2.2 SP-MAC: A Secure ProMAC Scheme

R2D2 provides the necessary building blocks for efficient ProMACs schemes that are resilient to network-level interference. One of its core features to achieve this protection is a shift from message dependencies to bit dependencies. As this shift fundamentally changes the interplay between tag aggregation and cryptographic operations, current ProMAC schemes (Whips [20], CuMAC [139, 140], and Mini-MAC [216]) cannot easily be retrofitted with R2D2's improvements. Therefore, we propose a novel ProMAC scheme for *staggered* progressive message authentication codes (SP-MAC) that leverages traditional and secure MACs (*e.g.*, `HMAC-SHA256` [29])

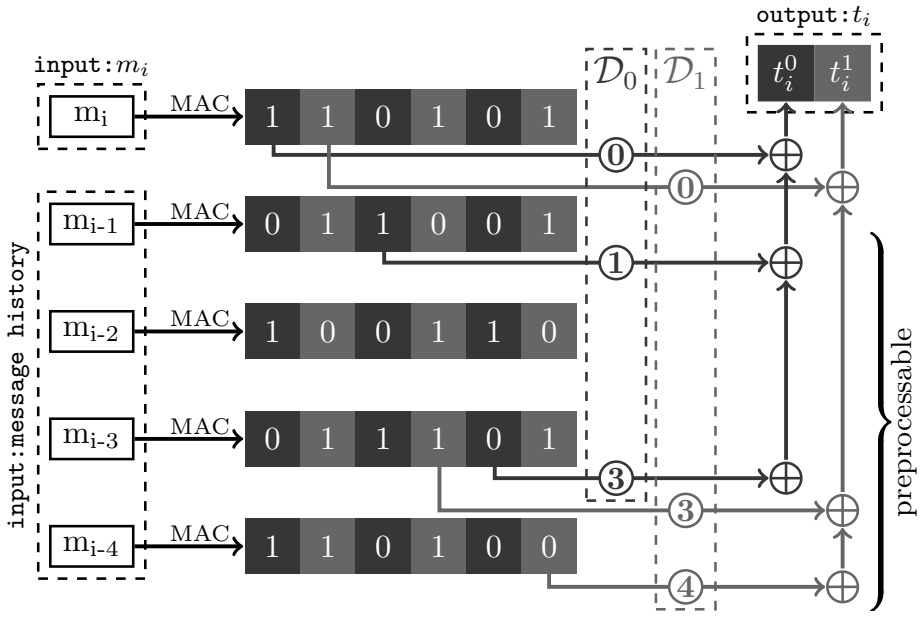


Figure 3.9 SP-MAC computes traditional MACs (only 6 bits shown here) for a message and derives aggregated and compressed tags through efficient XOR operations.

and aggregates these based on R2D2's secure dependency distribution using efficient XOR operations. As a result, SP-MAC not only provides built-in protection against network-level attacks but also achieves this as resource-consciously as existing ProMAC schemes. In the following, we introduce SP-MAC, discuss its security, and evaluate its performance.

3.2.2.1 Staggered Progressive MACs

Orienting ourselves on the good performance results of CuMAC [139, 140], SP-MAC, in contrast to Whips [20] and Mini-MAC [216], computes tags using an aggregation procedure for traditional MACs instead of defining how tags are directly derived from the recent message history. In a nutshell, SP-MAC thus operates as outlined in Figure 3.9, where exemplarily 6-bit traditional MACs are first computed and then compressed into 2-bit ProMACs tags.

In more detail, to generate t_i for a message m_i, SP-MAC first computes σ_i as a traditional MAC (*e.g.*, `HMAC-SHA-256-128` [29]) over m_i and a counter ctr, *i.e.*, $\sigma_i = \text{HMAC}(m_i, ctr)$. The counter is initialized with 0 and incremented after each message to protect against replay attacks. SP-MAC computes each bit of the tag t_i individually since each bit depends on a unique set of past messages (*cf.* Section 3.2.1.3). For pseudorandomly selecting these bit dependencies $\mathfrak{D}$, a pre-shared secret is used. Subsequently, SP-MAC derives t_i from σ_i and from the MACs of messages in the recent past as follows.

In the following, we refer to the j-th bit of t_i as t_i^j, *i.e.*, $t_i = t_i^0 \parallel \ldots \parallel t_i^{|t_i|-1}$, where $|t_i|$ denotes the bit-length of t_i. SP-MAC also splits σ_i into its individual bits, by first spitting σ_i into $\lceil |\sigma_i| / |t_i| \rceil$ parts. We then denote $\sigma_i^{a,b}$ as the b-th bit of the a-th

part of σ_i, *i.e.*, $\sigma_i^{a,b}$ is the bit at position $(a \cdot |t_i| + b)$ of σ_i. SP-MAC then computes each bit t_i^j of the final tag t_i of message m_i using the bit dependencies $\mathcal{D}_j$ for this bit as follows:

$$t_i^j = \bigoplus_{0 \le n < |\mathcal{D}_j|} \sigma_{i-\mathcal{D}_j[n]}^{n,j}$$

with $\mathcal{D}_j[n]$ representing the n-th entry of $\mathcal{D}_j$. At the start of a message stream, missing values are initialized with 0. Consequently, each bit t_i^j of t_i depends exactly on those messages defined in the corresponding bit dependencies $\mathcal{D}_j \in \mathfrak{D}$ and each bit $\sigma_i^{a,b}$ is included in exactly one tag.

To speed up calculations, SP-MAC partially caches previous messages' tags until they are fully depleted, *i.e.*, all bits have been incorporated into tag computations of subsequent messages. Furthermore, to reduce latency when computing the tag of a certain message, all but one XOR operation (incorporating the bit dependencies of this message into all tag bits at once) can be preprocessed since this processing only relies on bits from previous messages' tags. Using efficient XOR operations, which are mostly precomputable in idle time, SP-MAC is particularly suitable for resource-constrained environments, which are the prime profiteer of ProMACs.

3.2.2.2 Security Discussion

The security of SP-MAC follows from its resilience to key recovery attacks and the unforgeability of tags.

3.2.2.2.1 Resilience to Key Recovery Attacks

By overhearing the communication, an attacker learns strictly less information from channels that use SP-MAC for integrity protection than channels that rely on SP-MAC's underlying MAC scheme, as the tags computed by them are needed to compute the SP-MAC tag. Hence, a key recovery attack against SP-MAC is at least as hard as against the underlying MAC scheme. Thus, the key used to compute the tags cannot be recovered as long as the underlying MAC scheme does not expose a key recovery attack. The rationale behind this claim is the following: Given a stream protected with traditional MACs, an adversary can choose arbitrary bit dependencies (without needing access to the key) to derive the tags that would have been sent by SP-MAC, similar to the illustration in Figure 3.9. Thus, any key recovery attack against SP-MAC also attacks the underlying MAC protocol, as an adversary only needs to transform the underlying MAC into SP-MAC's representation before launching the attack.

3.2.2.2.2 Unforgeability of Integrity Protection

Security of (traditional) MACs relies on the unforgeability of tags, *i.e.*, attackers can neither directly forge tags nor guess the secret key. When considering ProMACs, the integrity of a single message is secured by multiple tags. At the same time, a

single tag protects multiple messages. Other than traditional MACs, ProMACs, therefore, have to define their security based on the (computational) infeasibility of circumventing the integrity protection of a single message.

We assume that the underlying MAC scheme provides a security level of l bits using an l-bit tag, *i.e.*, the probability of guessing a tag is not better than 2^{-n}. This assumption is expected to hold for common MAC schemes, *e.g.*, `HMAC-SHA-256-128` [29], and eases discussions on SP-MAC's security in the face of dropped packets. To show the security of SP-MAC, we first look at traditional MACs and think of an l-bit tag as n individual 1-bit MACs. Each of these 1-bit MACs provides 1 bit of security, *i.e.*, the probability that an attacker guesses it correctly is 2^{-1}. A message protected by a traditional MAC is transmitted with its n 1-bit MACs, and if it is not altered, all 1-bit MACs can be verified. For SP-MAC, this procedure changes as the 1-bit MACs are distributed over multiple packets and aggregated using XORs.

For pseudorandom MACs, this aggregation does not impact security, as XOR-ing multiple MACs still leads to a secure MAC scheme [31]. However, combining multiple MACs introduces dependencies on the successful reception of other messages, as a MAC can only be verified if *all* messages protected by the XOR-ed MACs were received unaltered. Here, R2D2 ensures that these dependencies are staggered and thus prevents the sandwich attack introduced earlier. Consequently, a significant number of messages have to be dropped to render a large number of MAC fragments covering a single message unverifiable. To illustrate this issue, in case R2D2 is parameterized for a maximum security loss of 16 bits per dropped message for a maximum security level of 128 bit and 4 targeted messages are dropped, SP-MAC still achieves a security level of at least 64 bit for the targeted message. However, SP-MAC's selected bit dependencies $\mathfrak{D}$ are derived from a pre-shared secret, thus hiding the strategy to achieve this worst-case attack from third parties. Consequently, an adversary needs to drop a suspiciously high number of transmissions [20] to even attempt to circumvent the integrity protection of a single message.

By providing resilience to key recovery attacks and ensuring the unforgeability of integrity protection, SP-MAC is able to realize secure progressive message authentication. Most notably, SP-MAC is the first ProMAC scheme that offers protection against network-level attacks, while still quickly achieving full integrity protection.

3.2.2.3 Performance Evaluation

ProMACs are specifically designed for resource-constrained environments, especially for wireless scenarios with high-frequency communication. In this context, energy consumption is generally a key metric for battery-powered devices. However, since the bulk of energy of those devices is consumed by the transmission and reception of wireless communication [222], the difference in energy consumption between ProMACs schemes is negligible. Nevertheless, for this reason, the overall energy cost of ProMACs is significantly lower compared to traditional MACs as the latter require longer tags and thus have a higher transmission overhead. Consequently, our evaluation focuses on the two most impacted aspects: the computational overhead of tag generation and the memory overhead to track message histories.

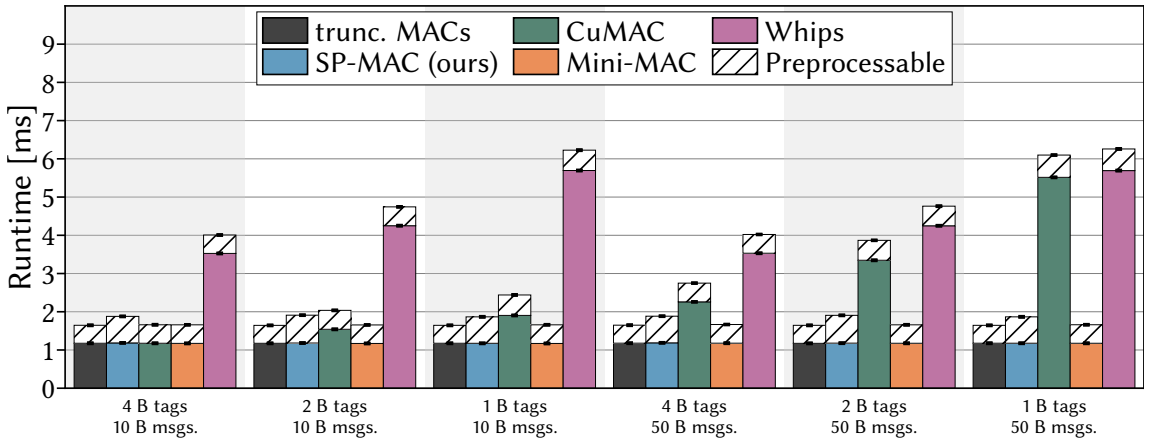


Figure 3.10 SP-MAC's tag computation adds only marginal (preprocessable) overhead compared to traditional MACs while thwarting network-level attacks against ProMACs.

3.2.2.3.1 Computational Overhead of Tag Generation

To evaluate the computational overhead of tag generation in SP-MAC, we implemented a prototype in C for the Contiki-NG [69] platform[1], which is widely used for low-power embedded systems in resource-constrained scenarios [101, 102, 114, 198]. Our prototype relies on the `HMAC-SHA-256-128` implementation of tinyDTLS as an underlying MAC scheme and uses the same R2D2 parametrization as in Section 3.2.1.3 (maximal security loss of 32 bits per dropped packet and 16-bit immediate protection for 4-byte tags).

To compare the performance of SP-MAC with state-of-the-art ProMAC schemes, we additionally re-implemented[2] Whips [20], CuMAC [139, 140], and Mini-MAC [216] for the same platform and underlying MAC scheme. Furthermore, we use a `HMAC-SHA-256-128` MAC as baseline reference (we truncate the MAC for a fair comparison, although the full MAC needs to be computed anyway).

With our implementations, we measure the time required to generate one tag of length 1, 2, and 4 byte, respectively, for 10 and 50 byte long messages on a Zolertia RE-Mote embedded device (ARM Cortex-M3 @ 32MHz, 16 kB RAM). All MAC schemes are parameterized for 128-bit security, except for the truncated MAC (baseline). We performed each measurement 30 times and report on the mean over these runs with 99% confidence intervals.

The measurements presented in Figure 3.10 establish a baseline of 1.17 ms for computing a traditional (truncated) MAC, irrespective of tag size and message length[3]. In contrast, the runtimes of Whips and Mini-MAC depend on the size of their internal state. For Mini-MAC, this state increases with growing message sizes and shrinking tags, ranging from 1.18 ms ms (4-byte long tag and 10-byte long message) to 5.52 ms (1-byte long tag and 50-byte long message). While Whips'

[1] `https://github.com/fkie-cad/spmac`

[2] Re-implementation was necessary as no source code was available.

[3] This baseline can further be improved if deemed necessary, *e.g.*, with hardware acceleration. Resulting performance savings carry over to all four ProMAC schemes.

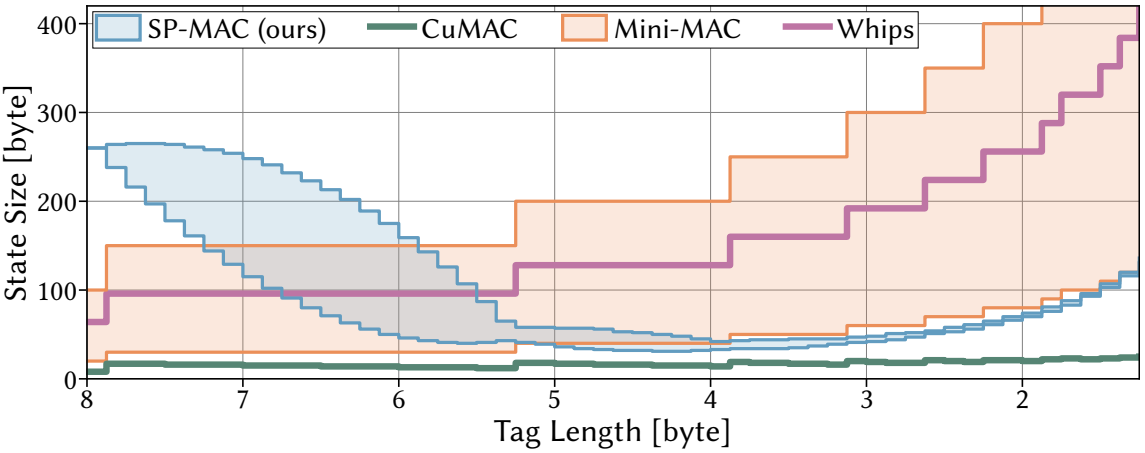


Figure 3.11 Despite tracking longer histories to thwart attacks, SP-MAC's memory footprint is in the same order as vulnerable ProMAC schemes. Note the decreasing x-axis that represents increasing space savings by ProMACs.

processing overhead is independent of message sizes, it also increases for shrinking tags from 3.52 ms for 4-byte long tags to 5.69 ms for 1-byte long tags. Whips starts with a higher processing overhead, as it calls the underlying MAC function twice, once to compute a new substate and once to derive the actual tag. On top of the baseline for truncated MACs, CuMAC introduces one additional non-preprocessable XOR operation, resulting in nearly identical runtime as truncated MACs. SP-MAC has the same online performance but adds a marginal 0.23 ms preprocessing overhead to protect against network-level attacks. As expected, neither SP-MAC's nor CuMAC's runtime is noticeably influenced by tag sizes or message length.

These results show that SP-MAC not only efficiently protects against network-level attacks but also operates at least as resource-conscious as existing ProMACs. Notably, SP-MAC's performance closely aligns with traditional MACs, showing that the benefits of ProMACs can be realized without increased latency and with only a minor increase in consumed processing power.

3.2.2.4 Memory Overhead for Keeping Integrity State

In contrast to traditional (truncated) MACs, ProMACs inherently have to keep state on past messages for the computation of future tags. To evaluate the impact of this state keeping on the memory consumption of resource-constrained devices, we evaluate the size of SP-MAC's state in relation to the tag lengths and, again, compare it to Whips, CuMAC, and Mini-MAC. To avoid potential bias due to implementation decisions, we conduct a theoretical memory analysis that is independent of our re-implementations of current ProMAC schemes. We parametrize all schemes to provide at least 128-bit security, with security loss limited by tag sizes, which is the worst case for SP-MAC w.r.t. memory overhead.

The results of our analysis are visualized in Figure 3.11. First, the memory overhead of Whips depends on the number of tags required for the desired security level (security level divided by tag length). Its overhead ranges from 64 bytes for 8-byte long tags

to 416 byte for 10-bit long tags. As the overhead of Mini-MAC additionally depends on the message length, we study the overhead for messages with lengths between 10 and 50 byte and find that the memory overhead of Mini-MAC follows the same trend as Whips. However, Mini-MAC's overhead depends on the message length, such that the resulting overhead spans an area around Whips' overhead. Exemplary, for a tag length of 4 byte, the overhead of Mini-MAC ranges from 40 byte to 200 byte. In contrast, CuMAC has a low and relatively constant memory overhead since the state stored per message decreases proportionally to the growing number of messages that are aggregated within one tag as tags vary in length.

Similar to CuMAC and Whips, the memory overhead of SP-MAC does not depend on the message length. However, it is influenced by the *random* selection of bit dependencies (*cf.* Section 3.2.1.3), as well as the interfering effects of tag length reductions (fewer bits per tag have to be precomputed) and the exponentially growing size of Golomb Rulers of increasing orders (tag precomputation has to start earlier). We thus display the range between the best and worst possible bit dependencies w.r.t. their memory overhead. As shown in Figure 3.11, this range is negligible for small tags and increases for larger tags because the number of short Golomb Rulers with two elements (used in tags longer than 42 bit) is limited.

Overall, the memory footprint of SP-MAC is well-manageable even for devices with scarce resources. Moreover, even in the worst-case scenario for SP-MAC in terms of memory overhead, its memory footprint is in the same order as current ProMAC schemes, all of which are vulnerable to the presented sandwich attack.

3.2.3 Summary

Progressive message authentication codes (ProMACs) promise the compression of authentication tags while preserving the strong security of traditional MACs by partly offloading integrity protection into the near future. Contrary to prior beliefs, we show that ProMACs cannot cope particularly well with lossy channels, preventing their deployment in many wireless scenarios: The generation and verification of tags depend on a sliding window of past messages, providing the foundation for our *sandwich attack*, in which the integrity protection of a whole message sequence is rendered void if merely two packets are dropped. Therefore, we consider it imperative to rethink how transmission failures influence the integrity protection of neighboring messages. With this in mind, we propose randomized and resilient dependency distributions (R2D2), which takes advantage of (i) optimal message dependencies, (ii) parameterized security guarantees, (iii) randomized bit dependencies, and (iv) optional immediate protection bits. Our evaluation shows that R2D2 significantly increases the resilience of ProMACs to lossy channels to unleash their full potential. At the same time, R2D2 achieves full integrity protection with comparable delays to current ProMAC schemes. To take advantage of R2D2 and realize a secure and resource-conscious ProMAC scheme, we propose SP-MAC that builds upon the proven security provided by traditional MACs, aggregating and distributing those across multiple messages using efficient XOR operations. SP-MAC is thus not only resilient to lossy channels

and sophisticated network-level attacks but also operates as resource-conscious as state-of-the-art ProMAC schemes.

3.3 MAC Aggregation on Lossy Channels

ProMACs are designed to provide immediate reduced message authentication that is improved over time. They are thus a special form of aggregated MAC [115] that aims to combine the authentication of multiple messages to save bandwidth. Instead of protecting the integrity of each message individually, a single authentication tag is responsible for protecting the integrity of multiple messages. In the following, we no longer focus on the intermediate achieved security guarantees but only on whether and when a message achieves full authenticity. Given a reliable channel, the resulting overhead can be reduced to a trade-off between saved bandwidth and the verification delay for received messages: Aggregating integrity protection of more and more messages reduces the overhead until it becomes negligible, but implies that the receiver has to wait for the reception of all messages affected by the aggregation before being able to check their integrity, resulting in significant delays.

The MAC aggregation schemes proposed over the years address weaknesses [71, 103, 122], split authentication tags over multiple messages [180] or provide progressive authentication [20, 139, 246]. While various implementations of security concepts, such as message authentication [226], have been evaluated and compared by the literature, such analyses of MAC aggregation schemes are practically non-existent. Most importantly, current evaluations of MAC aggregation schemes neglect that losing a single message from a set of messages with aggregated authentication tags may have cascading effects depending on the chosen MAC aggregation scheme. This phenomenon becomes increasingly relevant as more and more communication transitions to low-bandwidth wireless channels. MAC aggregation is thus arguably becoming more crucial for lossy channels than for its initial setting of reliable channels.

However, research, thus far, has not sufficiently addressed under which circumstances MAC aggregation on lossy channels is sensible and how to unlock its full potential. This knowledge is crucial to optimally utilize scarce bandwidth in wireless scenarios with an ever-growing number of participating devices. In the following, we analyze the performance of MAC aggregation for wireless channels. First, we look at general simulation to gain an understanding of the potential of MAC aggregation. Then, we evaluate this potential based on real-world scenarios, before integrating MAC aggregation into DTLS 1.3 and evaluating the scheme in a real-world testbed.

3.3.1 Existing MAC Aggregation Schemes

We first introduce the different sets of MAC aggregation schemes, grouped by their choice of dependencies $\mathcal{D}$ and computation of t^{agg}, the aggregated tag. We do, however, not focus on the exact aggregation function or the underlying MAC scheme, as those choices do not impact the scheme's susceptibility to packet loss. Under

these aspects, we present all four classes of aggregation that cover, to the best of our knowledge, all proposed schemes. For this presentation, we assume XOR-based aggregation with `HMAC-SHA256` as a suitable MAC scheme (including an appended nonce for replay protection).

`Trad`: To quantify the performance of existing MAC aggregation schemes, we compare them to the baseline performance of traditional MAC schemes. Therefore, we consider a traditional MAC scheme that authenticates each message m_i with an individual tag t_i. This computation thus solely depends on m_i, *i.e.*, $\mathcal{D} = \{0\}$. As we target 128-bit security, the HMAC-SHA256 is truncated to 16 byte.

`Agg(n)`: The most prominent scheme is aggregated MAC as introduced in 2008 [115] and later extended to prevent reordering attacks [71], allow messages to occur multiple times [122], and identify faulty messages in an aggregate [103]. For these schemes, a tag t^{agg} is only appended to each n-th message, where n is the parameter for how many messages' authentication tags are aggregated together. For our evaluation, we consider the aggregation of two, four, eight, and sixteen tags, *i.e.*, $n \in 2, 4, 8, 16$ to cover a range of different parameterizations. For every n-th message, a tag is then computed by XORing the authentication tags of all considered messages, as formalized in the following:

$$t_i^{\text{agg}} = \bigoplus_{i-n<k\leq i} t_k \quad \text{for } i \equiv 0 \pmod{n}$$

`Comp(n)`: As the tags computed by `Agg(·)` are too long for some applications, Compound MAC is proposed that splits across multiple messages [180]. Thus, each message carries a shortened authentication tag, the length of which is inversely proportional to the number of aggregated messages, *i.e.*, $|t| = {}^{128}/_{n}$. For our analysis, we again consider $n \in \{2, 4, 8, 16\}$. We formalize `Comp(·)` in the following, where $t[a : b]$ means the chunk from the a-th up to the b-th bit of tag t:

$$t_i^{\text{agg}} = \bigoplus_{\lfloor \frac{i}{n} \rfloor \cdot (n-1) \leq k < \lfloor \frac{i}{n} \rfloor \cdot n} t_k[(k \bmod n) \cdot |t| : ((k+1) \bmod n) \cdot |t|]$$

`SW(n,o)`: In classical sliding window-based ProMAC scheme each message is protected by a shortened tag that also verifies the integrity of the previous n messages. As `SW(·)` is not equipped to provide full security under packet loss, it can be compensated by additionally considering an overprovisioning factor o. This factor defines in percent how much security may be extended beyond the target, *i.e.*, $o = 100$ means that messages may have 256-security at the expense of longer tags as $|t^{\text{agg}}| = {}^{128}/_{n} \cdot (1 + {}^{o}/_{100})$. Here, we select a number of parameter combinations that perform best under various scenarios. The tag computation of `SW(·)` can be formalized as follows:

$$t_i^{\text{agg}} = \bigoplus_{i-n<k\leq i} t_k[k \cdot |t| : (k+1) \cdot |t|]$$

`R2D2(n,g,o)`: To address weaknesses of `SW(·)` in the presence of packet loss, `R2D2(·)` introduces dependencies that bound the effect a dropped packet can have on the

verifiability of any other message [246]. Therefore, the parameter g is introduced, which defines how much security any message loses at most if a surrounding packet is lost, *i.e.*, $g = 1$ in combination with 2 byte long packets means that any message can lose at most 16 bit of security. Furthermore, `R2D2(·)` randomizes the concrete dependency set $\mathcal{D}$ and assigns a different set to each bit of a tag. The final aggregate tag t^{agg} is thus a juxtaposition of bit-long tags t_j^{agg} and is defined as:

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

with $\mathcal{D}_j[n]$ representing the n-th entry of j-th bit dependency set $\mathcal{D}_j$.

3.3.2 Synthetic Measurements

We begin our analyses of MAC aggregation schemes by looking at synthetic measurements of simulated wireless channels. These measurements give us fine control over channel quality and payload length to investigate how these parameters influence the different MAC aggregation schemes. We first describe our setup before diving into the influence of channel quality and payload lengths. Finally, we look at the challenge of determining optimal payload lengths for given channel qualities under the additional constraint that the received data must be authenticated.

3.3.2.1 Simulation Setup

For our synthetic measurements, we use the network simulator `ns-3` (version 3.37), giving us fine-grained control over the underlying communication channel. As communication protocol, we choose the IEEE 802.15.4 protocol commonly used in constrained wireless environments, where we consider the most compact header of 5 byte. For payload lengths varying between 1 byte and the maximum supported 115 byte, we simulate the communication between two static antennas placed 25 m apart and extract binary loss traces of which transmitted packets have been correctly received. We additionally vary the transmit power between −21 dBm and −16 dBm using 0.1 dBm steps to increase the signal-to-noise ratio progressively, thus improving the channel quality and reducing PER. We only transmit each message once and neither implement acknowledgments nor retransmission, as these features are not available in all real deployments. For all combinations of transmit power and payload length, we simulated the transmission of 10 000 packets, of which we selected a random sequence of 5000 packets for each of the following analyses. In a standalone simulation, we then implement the behavior of the different classes of MAC aggregation schemes and their selected parameterizations to extract which messages eventually become authenticated for a given binary loss trace. Our measurements focus on authenticated goodput by the different MAC aggregation schemes, where authenticated goodput is the ratio of received and authenticated payload bytes (*i.e.*, excluding header and authentication tag) to the number of transmitted bytes. We initially focus on authenticated goodput as performance metric as it directly measures how efficiently

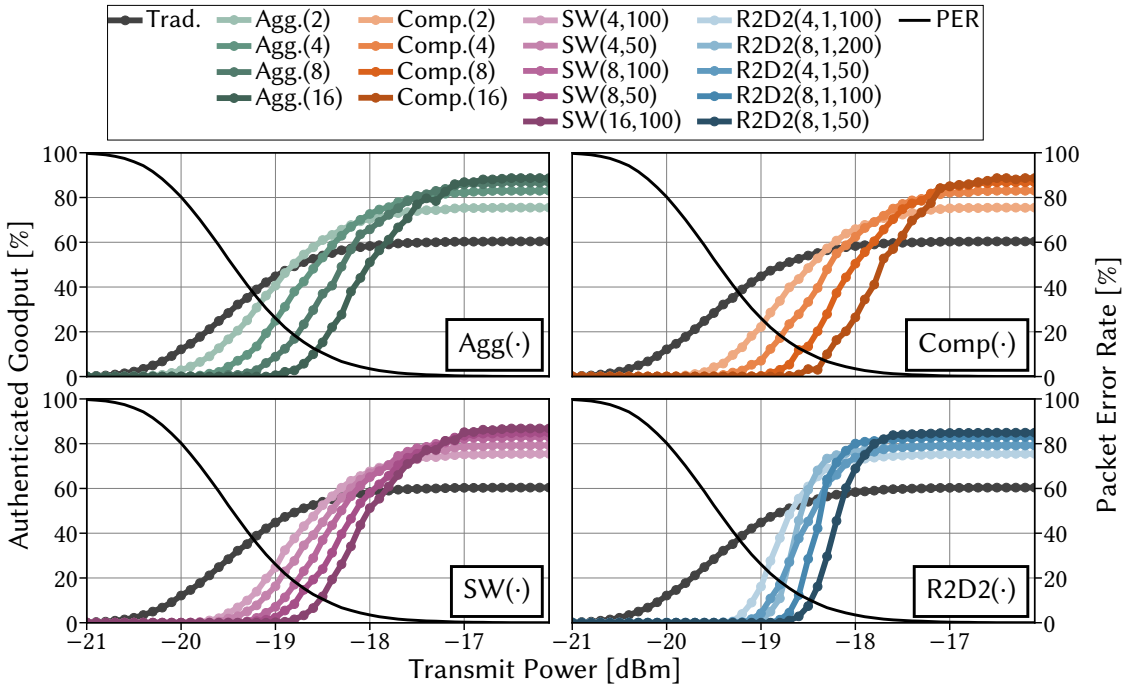


Figure 3.12 Traditional MAC performs best with high packet error, as all received data can be verified. For medium PERs, the aggregation of two tags with `Agg(·)` and various `R2D2(·)` parameterizations is preferable, while the aggregation of more messages with the simpler `Agg(·)` and `Comp(·)` schemes is desirable with low PERs.

the transmission channel is utilized, the improvement of which is the main goal of MAC aggregation.

3.3.2.2 Influence of Channel Quality on Goodput

For an initial understanding of the different MAC aggregation schemes, we fixed the payload length to 48 byte and gradually increased the transmission power, resulting in a slowly decreasing PER from 100 % to 0 %. Figure 3.12 shows our results.

We observe that all aggregation schemes exhibit the same general sigmoidal behavior: As the PER decreases, the achieved goodput increases slowly before increasing quickly and then leveling off. This behavior can be explained by the behavior of the packet delivery ratio (*i.e.*, the opposite of the packet error rate), which also increases first slowly and then rapidly as the channel quality improves. The interesting differences between the schemes and their parameterizations are thus defined by when and how goodput increases as the channel improves.

For the different aggregation schemes, we see that the maximally achieved goodput correlates inversely with the number of aggregated tags (parameter n). As a higher n results in, on average, shorter tags, a better maximal goodput can be achieved due to less overhead. Similarly, the transmit power where the goodput of the different schemes starts to take off also correlates with n. The increasing likelihood that at least one of the tags in an aggregate cannot be computed as the set of aggregated messages

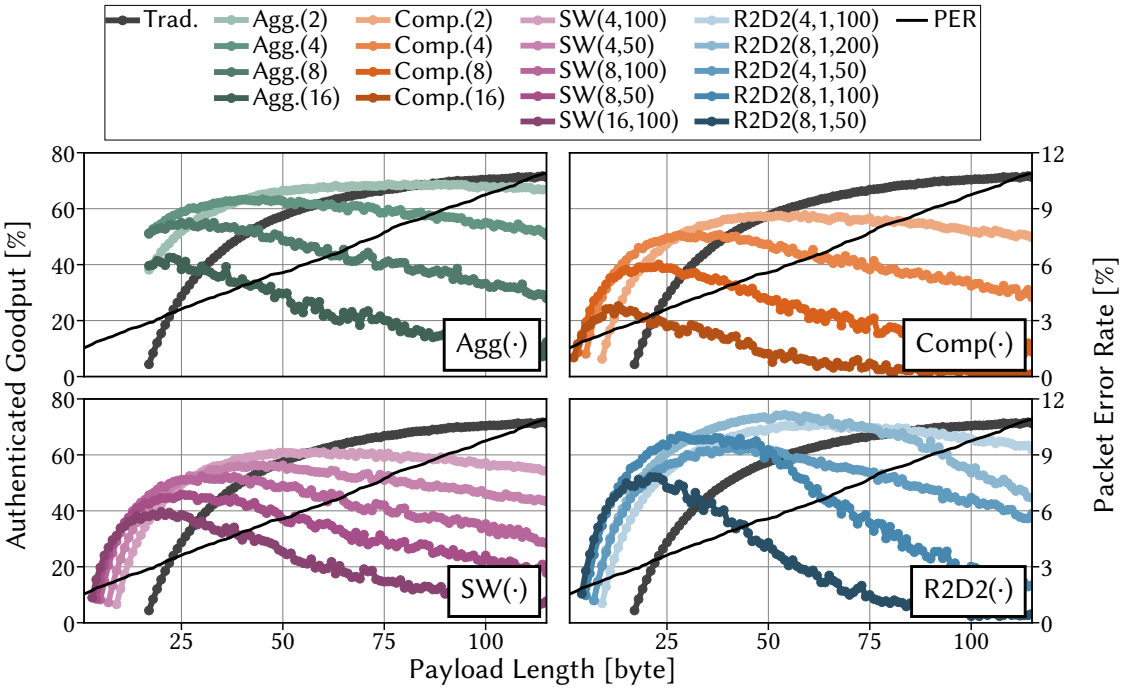


Figure 3.13 The PER increases for larger payloads, and the relative overhead of authentication tags decreases. Hence, the payload length influences which scheme performs best.

becomes larger can explain this observation. Thus, higher bandwidth-saving potential requires a better channel (*i.e.*, lower PER) to be beneficial over more conservative parameterizations. Consequently, traditional MACs perform best with high PERs while exposing the overall worst goodput as PER approaches 0 %.

Comparing the performance of the different aggregation schemes, we observe that all schemes tend towards the same discrete goodput dictated by their average tag length. However, the goodput provided by `Agg(·)`, `Comp(·)`, and `SW(·)` increases earlier but more slowly with increasing transmit power, in contrast to `R2D2(·)`, which suddenly jumps up once the channel is good enough. The behavior of `R2D2(·)` can be explained by ideally distributing the effects of packet loss to surrounding messages, such that if security levels for a few messages become good enough to consider the message authenticated, surrounding messages are close to the threshold as well. Overall, for transmit powers up −18.9 dBm (PER=18.5 %), traditional MACs perform best as they are not handicapped by the many lost packets. Then, the aggregation of two messages with `Agg(·)` is best until, between −18.3 dBm (PER=8.5 %) and −17.1 dBm (PER=0.4 %), there are different parameterizations of `R2D2(·)` that perform best. As the PER reduces further, the selected scheme becomes, however, less critical, and the differences for the same average tag length are marginal. Here, simpler schemes with no overprovisioning, such as `Agg(·)` and `Comp(·)`, are usually preferable. Consequently, it mostly depends on the channel quality, which aggregation scheme and parameterization achieve the best goodput.

3.3.2.3 Influence of Payload Length on Goodput

In Section 3.3.2.2, we consider a fixed payload length and slowly increase the transmit power to improve the signal-to-noise ratio. To better understand the behavior of the different MAC aggregation schemes, we now vary the payload length for a fixed transmit power of −18.3 dBm, where we have realistic PER between 1.5 and 10.9 % across the payload length range. We show our results in Figure 3.13.

With changing transmit power, we observe the same characteristics in the goodput curves of all aggregation schemes. Goodput first quickly increases before slowly dropping after reaching a maximum. This phenomenon can be explained by the overlapping effects of reduced relative overhead of authentication tags and growing numbers of unverifiable tags due to raised PERs with increased payload lengths. Thus, selecting the best MAC aggregation scheme depends on the underlying channel quality, packet lengths, and resulting variable PERs.

Furthermore, we can see that not all aggregation schemes can be employed for short payload lengths. Traditional MAC and `Agg(·)` append 16-byte authentication tags (to a fraction of all) messages and thus require payload lengths of at least 17 byte. The other aggregation schemes append a shortened tag to all messages, but the size of these tags also dictates how small messages can be. Thus, if transmitted packets can only carry a few bytes of payload, such as the unreliable CAN bus protocol, which supports at most 8-byte payloads and has no header fields intended for integrity protection, the choice of available MAC aggregation scheme shrinks.

Moreover, we observe different optimal payload lengths w.r.t. to authenticated goodput for the different schemes and parameterizations. While 114 byte payloads yield the optimal goodput of 71.7 % for traditional MACs, the overall maximal goodput of 74.4 % is achieved by `R2D2(8,1,200)` with a payload length of 54 byte. Thus, considering the combined impact of packet lengths and MAC aggregation under varying conditions is essential to identify optimal network configurations.

3.3.2.4 Optimal Packet Lengths for Authenticated Data

Prior results indicate that considering the MAC aggregation scheme is crucial when optimizing packet lengths for a given channel. This search for optimal payload length gathered interest in the past [13, 132, 211, 244] to make use of limited bandwidth availability or optimize the lifetimes of battery-powered devices. As resource-constrained devices consume most of their power for wireless transmissions [222], optimizing goodput is essential for improving device lifetimes. Assuming constant energy consumption for each transmitted bit at a given transmit power, the optimal combination of payload lengths and MAC aggregation scheme also optimizes device lifetimes. These packet length optimizations, thus far, only looked at received data and not received *and authenticated* data. Assuming the imperative requirement of authenticated data, we search for the optimal payload lengths to optimize goodput across varying channel qualities, considering the different MAC aggregation schemes. Our results are shown in Figure 3.14.

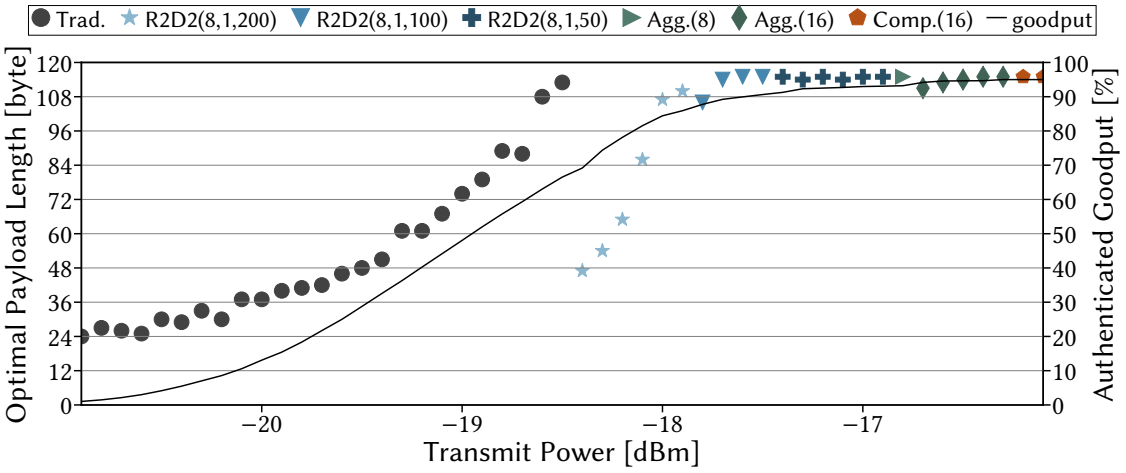


Figure 3.14 Different MAC (aggregation) schemes achieve a higher goodput as channel quality improves under optimal payload lengths. Unintuitively, changing a scheme can result in a reduced optimal payload length even if the channel improves.

For low transmission powers, *i.e.*, low signal-to-noise ratio, we see that traditional MACs,*i.e.*, no aggregation, performs best. This behavior can be explained by the initially high PER, even for small messages, such that aggregated tags have a high risk of being composed of at least one message that did not arrive. Here, the behavior observed in our setup matches related work [13,132,211,244] in that optimal payload lengths are initially short and then slowly increase as the transmit power is increased.

As the transmission channel improves, message aggregation starts to pay off since the benefits of shorter tags outweigh the risk of received data that cannot be authenticated. Here, initially, between -18.4 and −17.2 dBM, `R2D2(·)` under various parameterizations performs best. However, the best MAC aggregation scheme does not only change with better channels; the optimal payload also decreases on each change before slowly increasing again. Therefore, the optimal payload length for a transmit power of −18.5 dBM is 113 byte (with traditional MACs), but for a slightly higher transmit power of −18.4 dBM, it drops down to 47 byte (for `R2D2(8,1,100)`). We can observe this same phenomenon for other changes between MAC aggregation schemes, and it is more or less pronounced depending on the header sizes, where the static overhead of larger packet headers dampens the drop in optimal packet sizes.

Overall, we can see that the average tag lengths of the optimal schemes shrink for higher transmission power. Looking at the achieved goodput by the respective optimal scheme, we see a sigmoid curve that instead of leveling off at 82.5 % if only using traditional MACs, the different MAC aggregation schemes boosts this achievable goodput to 95.0 % as the PER approaches 0. However, it must also be understood that transmitting with optimal payload lengths is often not an option in practice. Here, the (established) applications and protocols often dictate the payload lengths, *e.g.*, a sensor may only have a single reading that should be transmitted quickly and thus has no other data to fill into the payload. Therefore, and because real wireless channels change over time, it is necessary to investigate MAC aggregation in real-world scenarios.

Scenario	Duration	Protocol	Header	Data	#pkts	PER	Src
ICS	8 hours	IEEE 802.15.4	11 B	20 B	57648	4.79%	[99]
Office	22 hours	BLE	10 B	32 B	79032	3.22%	[99]
Smart City (sta.)	131 days	LoRaWAN	13 B	16 B	18790	1.97%	[35]
Smart City (mob.)	250 days	LoRaWAN	13 B	24 B	17415	7.09%	[196]
Underwater	327 min	GUWMANET	31 bit	16 B	334	16.46%	[66]

Table 3.2 Limited bandwidth availability for integrity protection is a serious challenge across a wide range of lossy environments.

3.3.3 MAC Aggregation in Real-World Scenarios

Thus far, we have analyzed MAC aggregation schemes in controlled synthetic environments. While these analyses gave us insights into the behavior and nuances of the different schemes, they do not necessarily represent the entire story for realistic deployments. Here, we often have predetermined payload lengths dictated by available data or protocol specifications. Also, channel qualities vary dynamically over time, especially if some communication partners are mobile. In the following, we introduce distinct real-world scenarios, which we subsequently use to evaluate and compare the performance of the MAC aggregation schemes under realistic conditions.

3.3.3.1 Scenario Descriptions

For our realistic measurements, we rely on network traces collected from real-world scenarios. Each trace has constant payload lengths and transmission configurations, and we extract a binary loss trace of which transmitted packets have been correctly received or not. This trace is then fed into our simulation to analyze the MAC aggregation schemes. We summarize the scenarios in Table 3.4 and briefly introduce them in the following subsections.

Industrial Control System (ICS) Scenario. For the first scenario, we look at a measurement campaign of wireless communication in a 3600 m^2 production hall with nearly a billion transmitted packets [99]. We select a single representative link from the various configurations using the IEEE 802.15.4 protocol with a payload length of 20 B. Our trace covers a total of 8 h of traffic on a typical workday with an overall PER of 4.79 %. In this scenario, we observe primarily short bursts of packet loss with channel quality changing mostly over longer time windows (hours), while phases of high error rates (upwards of 50 %) are possible for several minutes.

Office Scenario. With the same measurement setup as for the ICS scenario, wireless links between nodes placed in various office rooms on a single floor have been measured [99]. Here, we select a Bluetooth Low Energy (BLE) communication link with 32 B payloads over a 22 h window during a workday. We observe a relatively constant error distribution with short error bursts of a few packets each and an overall PER of 3.22 %.

Smart City (Stationary) Scenario. Our first smart city scenario is based on the LoED dataset [35], where nine LoRaWAN gateways were placed in central London. We focus on the 18790 packets transmitted by a single stationary sender and received by any of the gateways. With an overall PER of 1.97 %, we see primarily isolated packet loss due to long idle times between two transmissions, and the channel only experiences long-term changes in quality over several days, potentially due to altering weather conditions.

Smart City (Mobile) Scenario. In this scenario, mobile LoRaWAN senders transmit to a total of nine stationary gateways for 250 days. Specifically, the sender was mounted to the top of a garbage truck driving through a 200 km^2 area in the city of Bonn [196]. We observe burstier errors and overall channel qualities changing significantly over days and weeks. The burstiness is likely due to the sender quickly entering and exiting the line of sight of a gateway, while the long-term changes presumably again relate to the weather conditions.

Underwater Scenario. Finally, we consider acoustic underwater communication, with a trace of 334 16-byte messages transmitted over 327 min between two stationary nodes placed in the sea [66]. This trace was collected during moderately rough weather conditions. Despite an overall high PER of 16.46 %, most of these errors occurred during long bursts interspersed with periods of high reception rates.

3.3.3.2 Evaluating MAC Aggregation in Realistic Scenarios

We now analyze MAC aggregation schemes in the different realistic scenarios introduced in the previous section. These scenarios are characterized by dynamic channels, differing communication protocols, and prespecified header and payload lengths. For each scenario, we analyze the goodput (*i.e.*, the amount of received and authenticated data) in Figure 3.15. We express the goodput as a percentage of the total amount of received payload data if no integrity protection was used. For the urban (static) and underwater scenarios, traditional MACs and `Agg(·)` cannot be included since the tags do not fit into the available payload.

We see different MAC aggregation schemes performing best in the synthetic measurements, depending on the investigated scenario. Payload lengths influence the concrete parameterization but do not directly correlate with which scheme performs best under the relatively small variations observed across the different scenarios. The best-performing MAC aggregation scheme thus depends primarily on two factors: overall PER and burstiness.

For the industry and urban (mobile) scenario, where overall PER is low and burstiness relatively high, `Agg(·)` performs best. During a burst, most packets in one set of aggregated messages are lost, while otherwise, most sets are received entirely and can be authenticated. For the office and urban (static) scenarios, `R2D2(·)` performs best due to the high PER and the short error bursts, where often only a single packet is lost. However, higher PERs do not immediately mean that `R2D2(·)` performs best (until traditional MACs are more favorable), as suggested by the synthetic scenarios. The long burst, where no traffic passes, in combination with a relatively good delivery

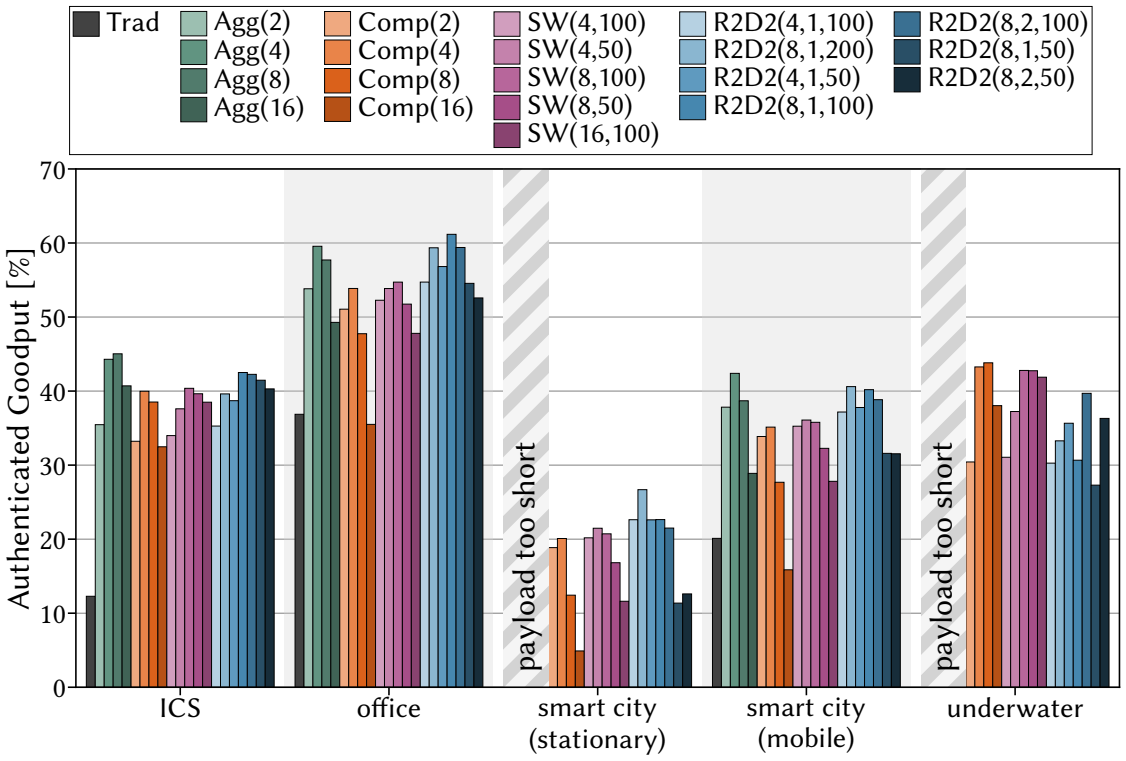


Figure 3.15 Different MAC aggregation schemes and parameterizations perform best, depending on the payload lengths, error burstiness, and overall PERs, such that the right scheme selection is non-trivial but crucial for the optimal use of constrained channels.

ratio otherwise mean that `SW(·)` and `Comp(·)` perform best in the underwater scenario. Overall, the best MAC aggregation scheme can achieve relative improvements of up to 24.2 % better goodput compared to the second best scheme.

However, more important than selecting the scheme is using the correct parameterization. If the wrong parameters are used for the best-performing MAC aggregation scheme, performance can drop by an average between 14.0 and 57.4 % in the worst case. With the PER of the different scenarios ranging between 1.97 and 16.46 %, we have parameterizations that result in average tag lengths of 4 byte performing best, which is more or less in line with the synthetic measurements from Section 3.3.2.2.

Overall, we see that PER and error burstiness play a significant role in finding the best scheme and parameterizations. Due to the relatively small variance in payload lengths across the scenarios, we, however, cannot confirm its low impact on selecting the best schemes. Nevertheless, we know that the potential gains achieved through MAC aggregation shrink with larger payloads. Ultimately, adequate parameterization is more important than finding the best MAC aggregation scheme, but both optimizations have a non-negligible effect on the achievable goodput.

3.3.3.3 Beyond Goodput as Evaluation Metric

Optimizing the goodput of MAC aggregation is the main goal for most scenarios. However, other effects must also be considered when choosing the MAC aggregation

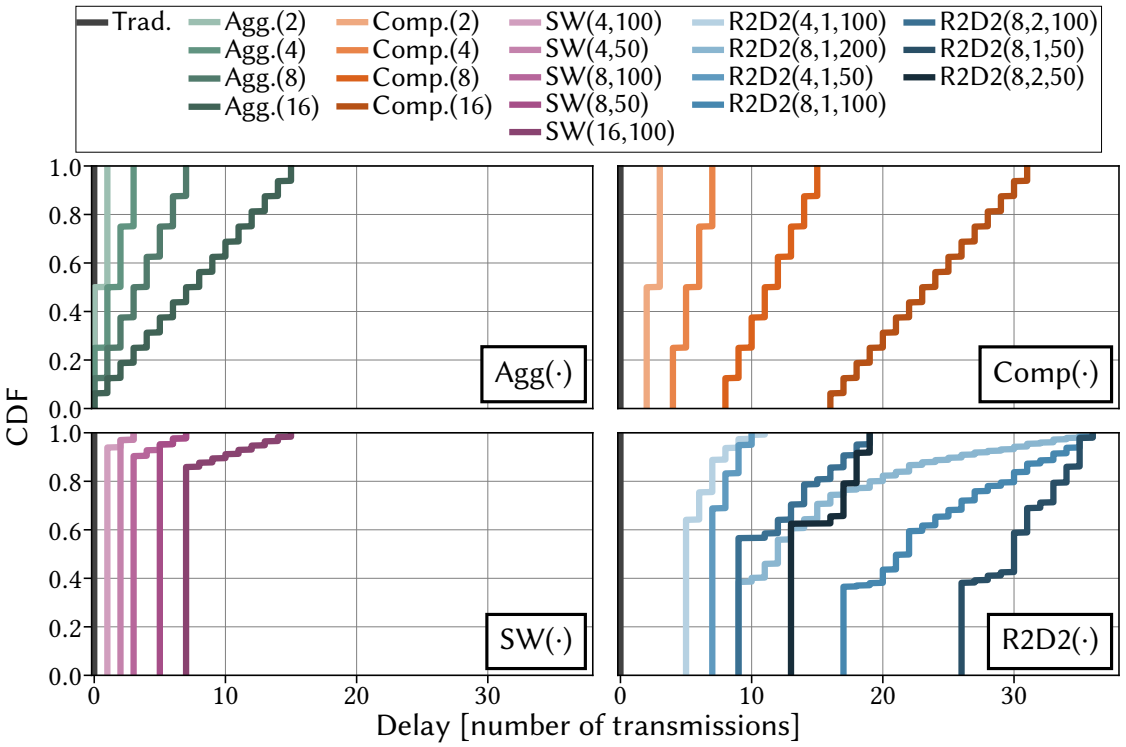


Figure 3.16 Verification delay is an inherent drawback of MAC aggregation. For scenarios where verification delay is critical, `SW(·)` does, however, provide highly consistent delays which control algorithms can thus anticipate.

scheme, such as verification delay, processing overhead, and susceptibility to jamming attacks. In the following, we compare the different MAC aggregation schemes w.r.t. these effects.

3.3.3.3.1 Average Delay until Authentication

First, we look at the authentication delay for the different MAC aggregation schemes. Traditional authentication tags can be verified immediately upon message reception, so no delay occurs due to waiting for additional data. With MAC aggregation, on the other hand, we need to wait until all messages depending on a specific tag have been received to verify it, which might introduce significant delays. To analyze these effects, we plotted the delay from the measurements on all traces from Section 3.3.3.1 as a CDF in Figure 3.16.

We see major differences in the behavior of the different aggregation schemes for these measurements. `Agg(·)` and `Comp(·)` periodically verify a set of prior messages together, such that a range of different delays occur with the same frequency. The concrete span of possible delay is then proportional to the parameter n of how many tags are aggregated together.

`SW(·)`, on the other hand, verifies messages continuously with an almost constant delay. This delay only varies if some messages get lost, which incurs a verification delay for surrounding messages. This behavior is beneficial for applications requiring periodic

messages with practically no jitter, *e.g.*, control algorithms in ICSs relying on a constant delay of the received information. R2D2(·) shows similar behavior for about half of all the received messages, while the rest have increasing delays. Again, the packet loss is responsible for higher delays, but since R2D2(·) distributed the effects of packet losses over multiple packets, more of them experience delayed verification. Moreover, the magnitude of these delays correlated with the overprovisioning factor o, allowing late authentication for messages that could otherwise not be authenticated.

In summary, the average delay until authentication of the different authentication schemes strongly differs. While Agg(·) offers, on average, the lowest delays, SW(·) has the most constant delays. On the other hand, R2D2(·) offers the best goodput for many scenarios with higher PER while messages have higher and more varying verification delays. Selecting the best aggregation scheme according to this delay thus depends on which balance the concrete application scenario demands between the goodput reduction and the type of verification delay.

3.3.3.3.2 Performance and Memory Overhead

Many of the considered scenarios involve resource-constrained IoT devices where substantial additional processing and memory overhead from the MAC aggregation scheme could significantly impact performance. Hence, we measure and compare the processing delay and memory overhead for tag computation and buffering by the different schemes. We conducted the analysis on the ARM Cortex M3 processor of a Zolertia RE-Mote board, a common choice to evaluate realistic resource-constrained hardware. As a baseline, we capture the time to authenticate a single 32 byte message with hardware-accelerated HMAC-SHA256, which is the underlying MAC scheme used for the aggregation schemes as well. We averaged the average processing times over 16 tag generations (not all schemes do the same computations for each message) and repeated this measurement 30 times. For the memory overhead, we measure the memory necessary to buffer tags before their aggregation, as all other memory overhead is implementation-dependent and is mostly optimized away by the compiler. The results of both measurements are presented in Figure 3.17.

Regarding processing times, we see only marginal overhead for all aggregation schemes except R2D2(·). There, we have a 168 to 237 % increase in processing times compared to the baseline, where the differences across parameterizations are mostly insignificant. This overhead stems from the bitwise processing of R2D2(·), which requires a significant amount of XOR and bitshift operations. This processing overhead is, however, mostly only impactful for applications that run on slower hardware and have tight latency requirements, especially considering that the sender *and* receiver must conduct this additional processing.

We see a different picture across the MAC aggregation schemes for the memory overhead. For Agg(·), Comp(·), and SW(·), the needed memory depends on the message history. More tags must be stored concurrently for shorter aggregated tags, resulting in higher memory overhead. Here, the bitwise processing of R2D2(·) helps to partially process tags when new messages arise. Consequently, the memory depends mainly on the overprovisioning factor and less on the number of aggregated tags.

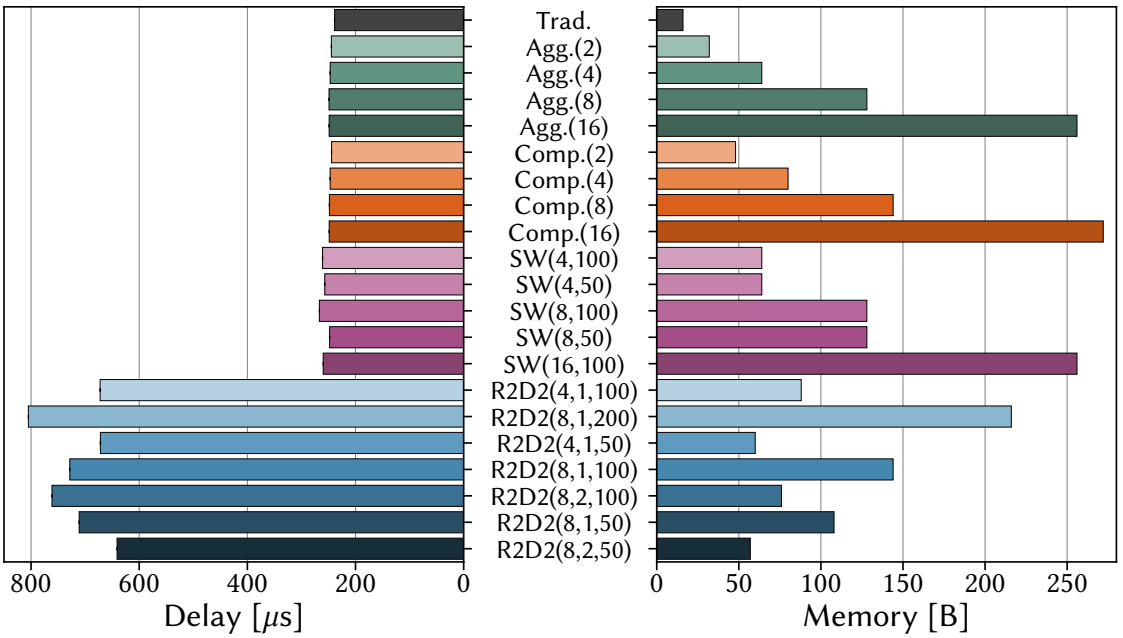


Figure 3.17 Only R2D2(·) introduces significant processing overhead over traditional MACs. Memory overhead, on the other hand, is mostly dependent on how many tags are aggregated together, but it is so small that it should rarely be a decisive factor.

The magnitude of the required additional memory for MAC aggregation schemes is, however, small enough that it should rarely influence the decision on which scheme should be deployed.

3.3.3.3.3 Resilience to Adversarial Interference

In our final analysis, we compare the resilience of MAC aggregation schemes to selective jamming attacks. Selective jamming refers to jamming specific messages to prevent their correct reception, which enables stealthy and energy-saving attacks as dropped packets are hardly distinguishable from random packet loss [19, 257]. In the context of MAC aggregation schemes, a sophisticated attacker can amplify the effects of a denial-of-service attack due to the employed MAC aggregation. For example, for Agg(16), it suffices to jam every 16^{th} packet to reduce the (authenticated) goodput of the channel to zero.

For our measurements, we considered the trace from the urban (mobile) scenario introduced previously, as its payload is large enough for all schemes, and urban settings provide easy access to potential attackers. For each aggregation scheme, we developed the optimal jamming attack strategy to minimize the goodput at the receiver. In Figure 3.18, we show how the achieved goodput of the different aggregation schemes is impacted by increased attacking capabilities.

The x-axis represents the number of overall dropped packets in percent on a logarithmic scale. For traditional authentication, we see, as expected, that the channel can still transmit authenticated data as long as not the entire channel is jammed. In general, we note that shorter average tags are more susceptible to selective jamming

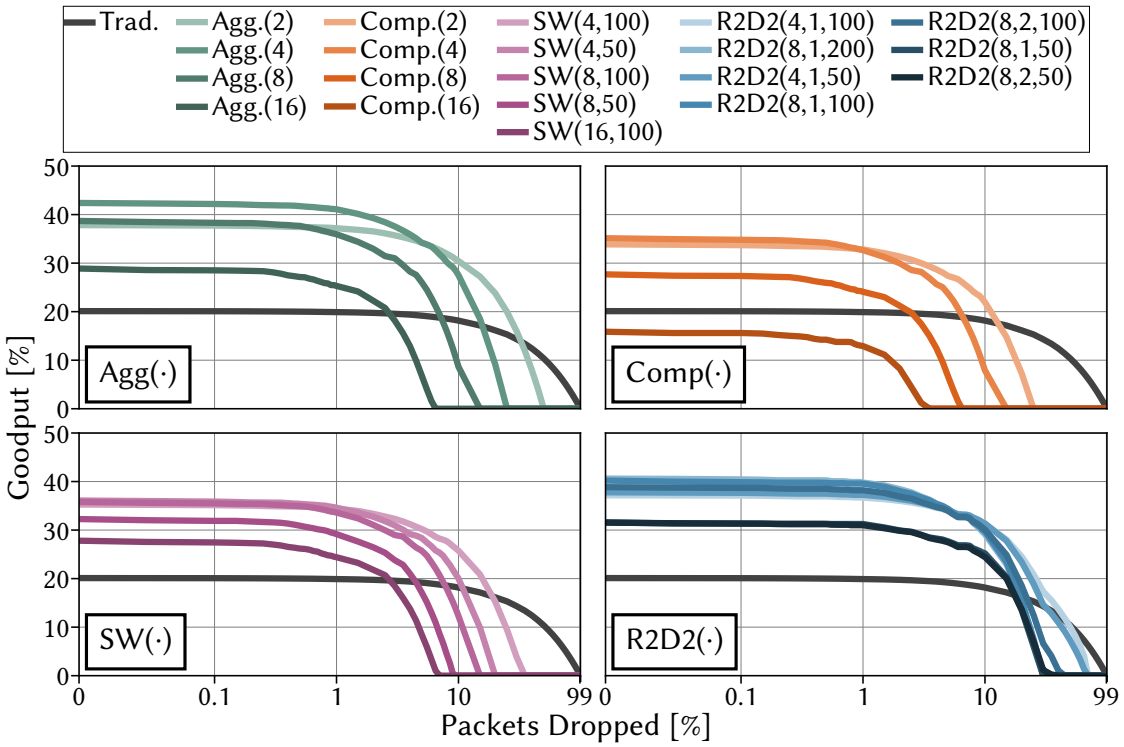


Figure 3.18 `R2D2(·)` shows significantly increased resilience to denial-of-service attacks through selective jamming, especially if an attacker jams less than 10 % of messages to remain stealthy or conserve energy.

attacks, as each tag requires many received messages to become verifiable. Considering the shortest tags ($n = 16$) for `Agg(·)`, `Comp(·)`, and `SW(·)`, we see that dropping between 27 and 29 % targeted packet already suffices to prevent all data transmission over the channel.

The behavior of `R2D2(·)` requires, however, a separate analysis since one of the protocol's design goals is resilience against jamming attacks. Therefore, the exact dependencies between tags and messages are kept secret, such that attackers can only design their strategy to inflict the most damage for the average dependency selection. Furthermore, the design of `R2D2(·)` explicitly distributes the effects of packet losses (malicious or not) over many packets, thus cushioning the impact of selective jamming. Hence, up to 15 % of packets need to be dropped to reduce goodput by even 20 %. However, once a critical mass of packet loss occurs, such a distribution no longer suffices for compensation, and the goodput quickly drops.

Overall, we can say that `R2D2(·)` is the most resilient scheme in the presence of a selective jammer. Considering our entire analysis, no scheme is an outright winner, and each scheme has its benefits. To summarize these findings, guide operators toward the right MAC aggregation scheme, and identify open research questions, we provide general recommendations in the following section.

3.3.3.4 Guidelines on Employing MAC Aggregation

In general, MAC aggregation shows promising potential to boost available bandwidth on lossy channels for various scenarios. However, not every scenario benefits from MAC aggregation compared to traditional MACs. More importantly, choosing the correct scheme and parameters is decisive in answering the questions of *when* and *how* to use MAC aggregation. Therefore, in the following, we deepen this discussion by providing general guidelines on employing MAC aggregation based on our empirical measurements.

3.3.3.4.1 When to Use MAC Aggregation on Lossy Channels?

From our analysis, it is evident that MAC aggregation reliably improves goodput for relatively high PERs of 10 % or below. In cases where the PER is higher, it is often more beneficial to rely on traditional MACs or, at most, aggregate MACs for no more than two messages (*i.e.*, setting the parameter n to 2). However, for high PERs due to long error bursts where hardly any traffic arrives, MAC aggregation can still be beneficial (*cf.* Section 3.3.3.2).

Furthermore, we investigated the relationship between payload lengths and the resulting benefits of MAC aggregation. For instance, in scenarios involving 200 byte payloads and minimalistic 5 byte headers, a MAC aggregation scheme aggregating 16 tags (*i.e.*, $n = 16$) could still generate a 7.3 % goodput improvement. Consequently, we conclude that MAC aggregation, in general, offers the most substantial benefits for short payload lengths, up to a few hundred bytes, and moderate PERs of up to 10 %. As substantiated by the real-world scenarios (*cf.* Section 3.3.3.1), this is precisely the kind of communication that occurs in many (industrial) Internet of Things (IoT) scenarios, leading to the question of how to use MAC aggregation in such scenarios to gain the most benefit.

3.3.3.4.2 How to Employ MAC Aggregation on Lossy Channels?

In our evaluations of the goodput improvements that different MAC aggregation schemes and parameterizations can bring in real-world scenarios (*cf.* Section 3.3.3.2), we have seen that no aggregation scheme is a clear-cut winner (even when solely focusing on goodput as an evaluation metric). Moreover, we have seen that the correct parameterization for a given scenario is crucial to achieving optimal performance. These observations thus warrant a more nuanced discussion of when to use which MAC aggregation scheme and with which parameters.

Focusing solely on goodput, we see that generally `R2D2(·)` achieves the highest performance for PER between 0.4 and 8.5 %, especially when packet errors occur as short bursts. For lower PERs and traffic with longer error bursts, the better performance and simplicity of `Agg(·)` is often preferable. If the periodic 16-byte tags for `Agg(·)` are not supported by the application (*e.g.*, due to fixed message sizes), `Comp(·)` is a good alternative to realize a constant tag size across all messages. Considering the parameterizations, a high n has the potential to realize better

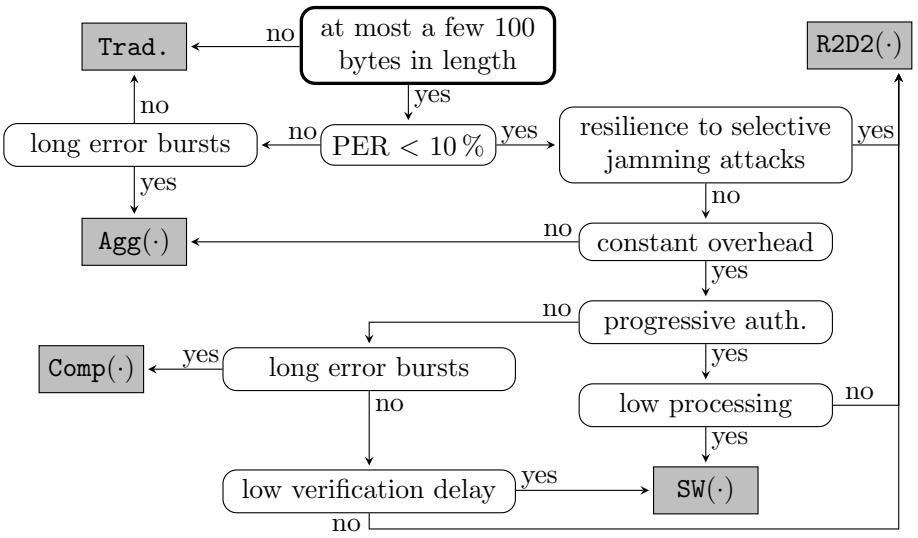


Figure 3.19 The optimal MAC aggregation scheme depends on many different characteristics. This decision diagram assists in this selection process.

goodput, but only if the PER is relatively low. For the overprovisioning factor o of SW(·) and R2D2(·), 100 is usually the best or at least a decent choice. R2D2(·)'s g-factor is best set to 1 in those scenarios where R2D2(·) achieves the best goodput. Overall, SW(·) rarely outperforms the other schemes if only considering goodput since it is not designed for lossy communication [246]. Nevertheless, it can still be a sensitive choice when also considering *e.g.*, verification delays.

One disadvantage of MAC aggregation compared to traditional MACs is the inherent verification delay, which we investigated in Section 3.3.3.3.1. This delay occurs as most messages cannot be verified directly upon reception and thus need to be buffered or processed optimistically [246], *i.e.*, processed under the assumption of being genuine before full integrity verification. This risk can be reduced by the two progressive schemes SW(·) and R2D2(·), already providing some, yet reduced, security guarantees immediately upon message reception. Furthermore, if an application requires complete message verification, SW(·) provides deterministic verification delays, beneficial for real-time control.

Concerning other potential dimensions for selecting the best MAC aggregation scheme for a given scenario, memory overhead is so small that it should rarely be a decisive factor. When interested in optimizing processing overhead, only R2D2(·) shows a clear disadvantage (*cf.* Section 3.3.3.3.2) compared to the other aggregation schemes. Finally, if resilience to denial-of-service attacks through selective jamming is essential, R2D2(·) shows clear advantages over the other schemes. However, if another scheme must be used (*e.g.*, due to the excessive processing overhead of R2D2(·)), then lowering the parameter n can reduce the effects of attacks at the cost of reduced goodput under normal operation.

3.3.3.4.3 Selecting a MAC Aggregation Scheme

We observe that often many dimensions must be considered to decide when and how to perform MAC aggregation. To help operators in their decision process, we provide two forms of assistance. First, we provide a decision diagram to select the right MAC aggregation scheme in Figure 3.19 based on basic network characteristics and feature demands. Secondly, and for more detailed analysis, we provide an evaluation tool[4] to aid further in this decision process. Our evaluation tool takes as input the header and payload lengths as well as an example binary loss trace, *i.e.*, a series of 1s and 0s for received and dropped packets, respectively. It provides a comparison of all MAC aggregation schemes and their parameterizations (as analyzed in this thesis) for the given scenario. In combination with these tools, our guidelines support operators in deciding when and how to employ MAC aggregation and help researchers to identify further opportunities to optimize existing MAC aggregation schemes.

3.3.4 Summary

While MAC aggregation is a well-known technique to save bandwidth overhead on reliable channels, its potential for wireless communication has not been previously studied in detail. We show that different MAC aggregation schemes, including ProMACs, yield strong goodput improvements even on such lossy channels. Here, different aggregation schemes bring unique benefits such as short verification delays or resilience to selective jamming attacks. Our simulations indicate that, in general, MAC aggregation is particularly effective in scenarios with PERs below 10 % and for payloads shorter than a few hundred bytes.

3.4 Integrating MAC Aggregation into DTLS 1.3

MAC aggregation promises to save valuable bandwidth in resource-constrained networks by shifting integrity protection from single to multiple packets. Nevertheless, before these promising results can be applied in the real world, two additional fundamental challenges must be solved. First, we need to understand how MAC aggregation can be integrated seamlessly 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 hardly reflect the dynamic channels seen in, *e.g.*, manufacturing environments, it is crucial to assess the true potential of MAC aggregation in the real world. Therefore, holistic evaluations of a complete communication protocol stack (to *e.g.*, fully consider the overhead of each layer) under realistic channel behavior on actual hardware are required.

To the best of our knowledge, we are the first to investigate the applicability of MAC aggregation over lossy channels in a physical deployment. Therefore, we integrate

[4] `https://github.com/fkie-cad/mac-aggregation-analysis-tool`.

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 integrating 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 between sender and receiver. This design enables the use of MAC aggregation with various existing and future cipher suites. Moreover, we show how dynamic parameter updates allow adaptations 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 message authentication [20]. Overall, we show that MAC aggregation can be seamlessly integrated into DTLS 1.3 without breaking backward compatibility and that goodput can be significantly improved. At the same time, energy and bandwidth usage are decreased in real-world deployments. 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.

3.4.1 Considered MAC Aggregation Schemes

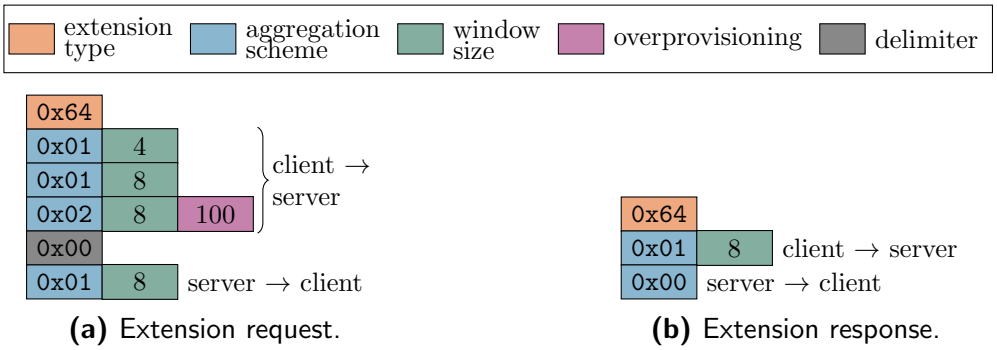
Previously, we saw a wide range of potential MAC aggregation schemes which can be described by a dependency set, *i.e.*, which messages are protected with one tag, and an aggregation function, *i.e.*, how the tags are aggregated together. In the following, we focus on two schemes, `Agg(·)` and `R2D2(·)` (and `Trad` as baseline), with XOR as their aggregation function and dependency sets that focus on different features (*cf.* Section 3.3.1). `Agg(·)` minimizes verification delays while `R2D2(·)` thwarts selective jamming attacks. For `R2D2(·)` we omit the parameter g and fix the parameter n to 8, as this yields the best results in our prior experimentation. While we use XOR as aggregation function due to its efficiency, other provably secure aggregation functions, such as the one used by Whips [20], 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 [71].

3.4.2 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 it connects to supports MAC aggregation and 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

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

Table 3.3 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.

Figure 3.20 During a DTLS 1.3 handshake, the client can append an extension to request MAC aggregation. Figure 3.20a shows the structure of this request. It contains a list of supported aggregation schemes and parameters for both communication directions. Figure 3.20b depicts the response that selects the concrete schemes and parameters for the session. For server-to-client communication, the selection of 0x00 indicated that no MAC aggregation is wanted.

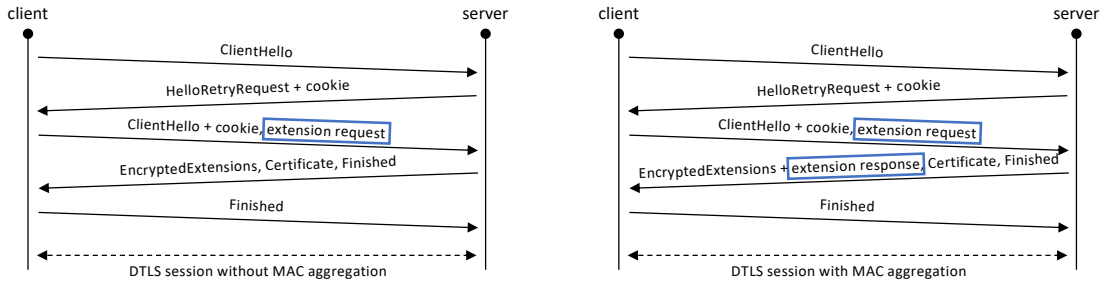
scheme is represented by a unique 1 byte identifier as shown in Table 3.3. Each aggregation scheme is defined by its dependency set and aggregation function.

The structure of the extensions is displayed in Figure 3.20. 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, first from client to server and then from server to client, separated by a null byte.

The Extension is initially sent by the client 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 [202] and a conventional DTLS 1.3 session is established as illustrated in Figure 3.21a.

Otherwise, Figure 3.21b 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 desirable security levels 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 CBC-MAC is appended.



(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.

Figure 3.21 The extension mechanism of DTLS 1.3 allows the seamless deployment of support for MAC aggregation alongside devices not supporting the extension.

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.

3.4.3 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 byte. Consequently, authentication tags no longer have a constant length, and ciphertexts may be theoretically as short as 2 byte (an encrypted `content type` with 1 byte 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.

3.4.3.1 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. 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

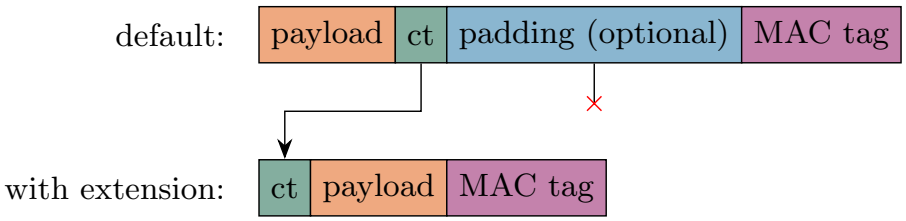


Figure 3.22 After agreeing on MAC aggregation, DTLS records carry the content type as the first field to make it easily recoverable during decryption. Consequently, MAC aggregation must be used in combination with a stream cipher, *e.g.*, AES in counter mode, as the content type no longer acts as delimited between the payload and padding.

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

3.4.3.2 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 or 16 byte 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 or 16 byte, MAC aggregation can lead to ciphertexts that are as short as 2 byte. DTLS 1.3 proposes to pad too short plaintext 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 byte 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 listening 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.

3.4.4 Upper Layer Interface

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

3.4.4.1 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 corresponding data can be silently discarded or answered with a fatal error, depending on the receiver's configuration. Moreover, buffered data is eventually discarded if data missing for verification is considered definitively lost.

3.4.4.2 Optimistic Data Processing

The idea of progressive message authentication is to optimistically process data before full integrity verification takes place [20]. 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 ICSs, are envisioned to benefit from such optimistic security [51, 231]. With MAC aggregation, such an optimistic operation can be supported.

Therefore, the receiver needs to set a security level threshold upon which data is passed to the upper layer and a callback function that is called 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 should lead to a fatal error that closes the communication channel, is scenario-dependent.

3.4.5 Dynamic Aggregation Parameters

While the aggregation scheme is fixed for a session, one may want to dynamically change 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.

3.4.5.1 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 3.20.

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 process it immediately upon reception. All subsequent messages 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.

3.4.5.2 Acknowledging AggregationUpdates

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(·)`, 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(·)`, 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(·)` with adapted overprovisioning, starts.

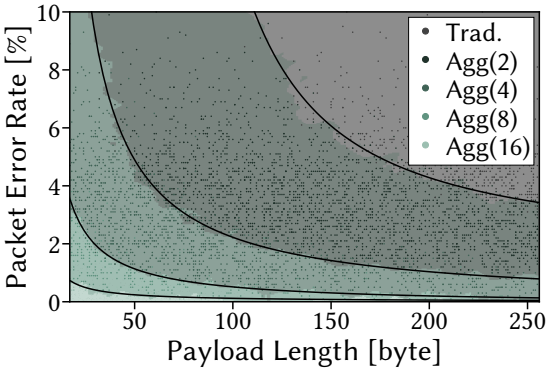
3.4.5.3 Parameter Selection

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.

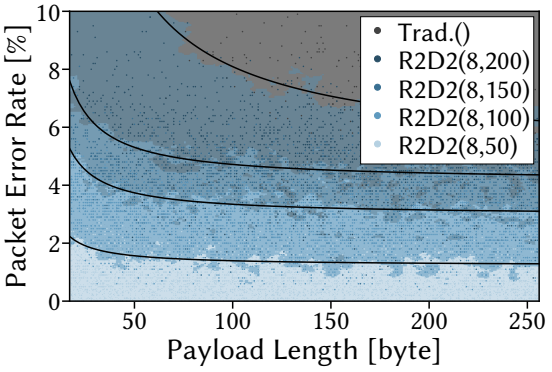
We want to understand how the schemes perform under different channel conditions. Therefore, we generated binary packet loss traces for 10 000 channels based on the Gilbert-Elliot model based on real-world measurements [99]. For each of 10 000 sampled traces with varying packet lengths, we check which 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 3.23.

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

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 perform 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 3.23. 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



(a) Aggressive MAC aggregation with `Agg(·)` only performs best 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 %.

Figure 3.23 We approximate the optimal aggregation parameters, *i.e.*, those yielding the highest goodput, based on boundaries described by formulas of the form $y = a \cdot e^{b/x} + c$, where x is the payload lengths, y is the current PER, and a, e, c as well as b are parameters.

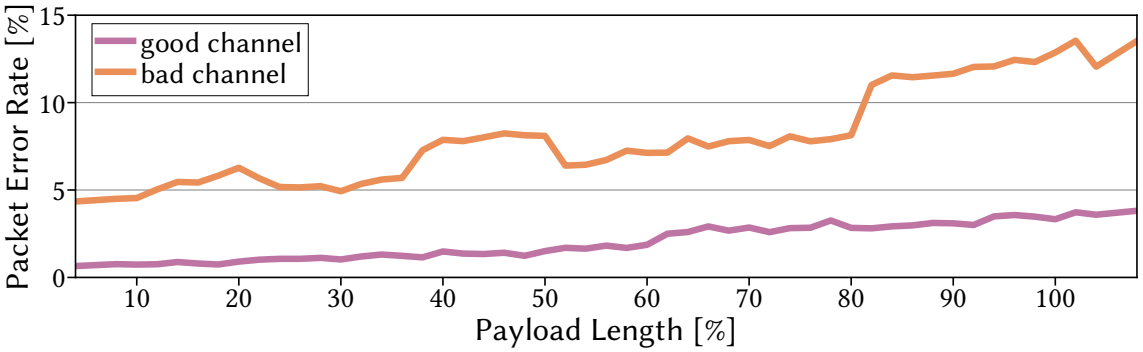


Figure 3.24 As expected, the PERs increase for longer payloads as the probability of uncorrectable bit flips increases.

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 3.23. 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.

3.4.6 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.

3.4.6.1 Measurement Setup

We conduct our measurements on two Zolertia RE-Mote boards equipped with a Cortex M3 @ 32 MHz, 32-bit CPU, the same hardware as used before. We also use the Contiki-NG operating system but adopt the new wolfSSL DTLS 1.3 implementation (instead of DTLS 1.2) to support MAC aggregation, as DTLS 1.3 introduces optimization to save bandwidth for header compression and thus nicely complements 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.

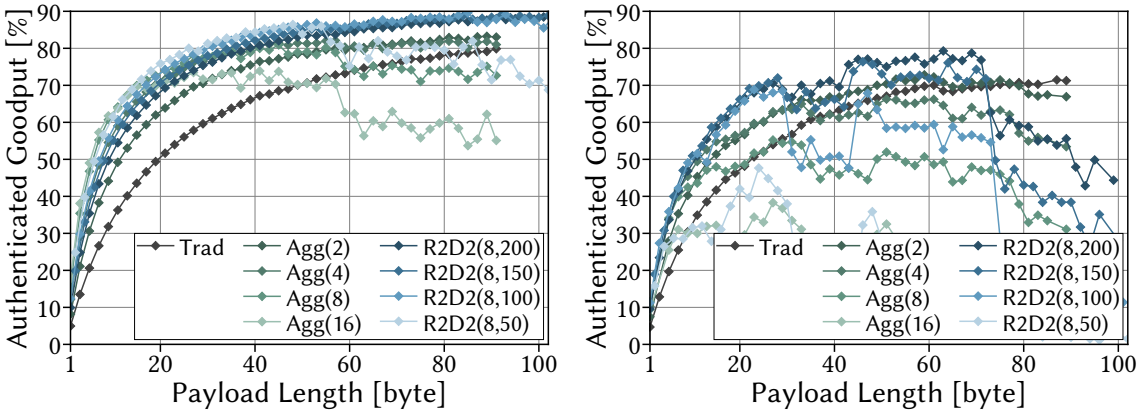
3.4.6.2 Aggregated MACs vs. Longer Packets

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 byte and 108 byte over a good and a bad channel. We plot the resulting PER in Figure 3.24. 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 the scenarios are evaluated individually.

3.4.6.2.1 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 3.25.

On the good channel in Figure 3.25a, MAC aggregation significantly improves authenticated goodput. Short payloads observe the most significant gains, which is expected as the overhead of MACs is proportionally larger. Moreover, `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. 3.24). 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 byte vs. up to 104 byte).



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

(b) With higher PER, an appropriate MAC aggregation scheme still yields goodput improvements of up to 146 % for 1 byte payloads.

Figure 3.25 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.

On the bad channel in Figure 3.25b, 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 3.4.5.3. Overall, we still see, for example, a 38 % goodput improvement by `R2D2(8,200)` for 20-byte payloads.

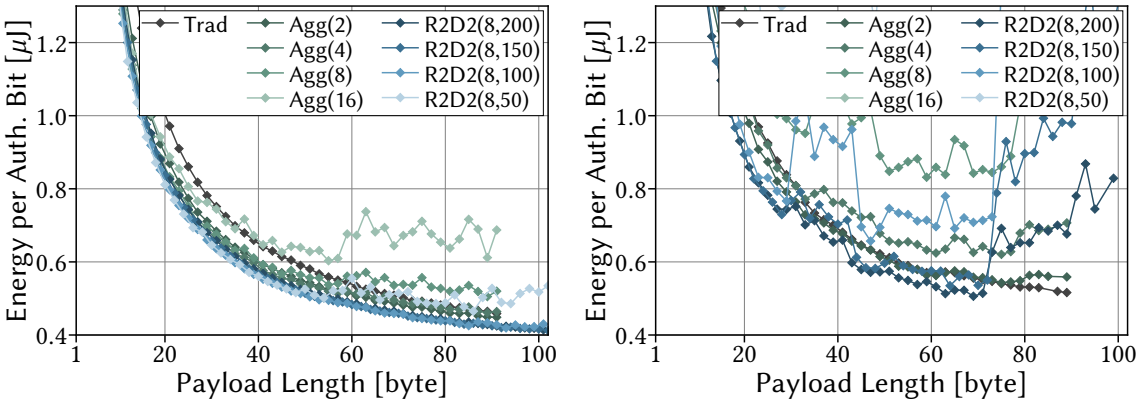
We conclude that MAC aggregation indeed improves authenticated goodput. However, if the aggregation is too aggressive for a given channel, gains quickly diminish.

3.4.6.2.2 Energy Consumption

In resource-constrained scenarios, the longevity of battery-operated devices may also be a critical concern. We base the energy cost in Figure 3.26 on measurements of the energy cost of transmitting packets of varying sizes, with expected linear increase from around 0.6 µJ for 4-byte packets to around 2.6 µJ for 110-byte packets.

We observe that MAC aggregation saves energy for all payload lengths on the good channel in Figure 3.26a. The best improvements occur for small payloads, with energy saving of 27.1 % for 10-byte payloads by `R2D2(8,50)`. But even the energy cost for longer payloads is reduced. For example, the energy per authenticated bit is reduced from 0.46 µJ to 0.42 µJ (8.4 %) by using `R2D2(8,150)` for the maximum supported 91-byte payload. Meanwhile, as `R2D2(8,150)` supports up to 102-byte payloads, energy savings of 10.1 % can be achieved even for long payloads.

On the bad channel in Figure 3.26b, we see that energy saving closely correlates with goodput improvement. Here, we see energy savings of up to 20.6 % for 21-



(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.

Figure 3.26 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.

byte payloads with `R2D2(8,150)`. However, for longer payloads above 70 byte, even `Agg(2)` is outperformed by traditional MACs.

MAC aggregation can thus realize significant energy saving unless payloads are long and PERs are high. To achieve these benefits, the aggregation scheme and its parameters must, however, be appropriately chosen.

3.4.6.3 Static Aggregation Parameters

Now, we look at the performance of MAC aggregation based on three concrete scenarios. We will look at the achieved goodput and compare it to the results of Figure 3.25a, before looking at the authentication delays of the different schemes.

3.4.6.3.1 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 3.4. 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, our evaluation ran for 1 hour, for a total of 27 hours, over a quiet weekend. The grey diamonds on the second y-axis 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.

3.4.6.3.2 Goodput

We show the achieved goodput in Figure 3.27. These results validate the methodology from Section 3.4.6.2 and Figure 3.25. Thus, MAC aggregation can indeed improve

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

Table 3.4 Settings of the three scenarios evaluated in more detail for Section 3.4.6.3.

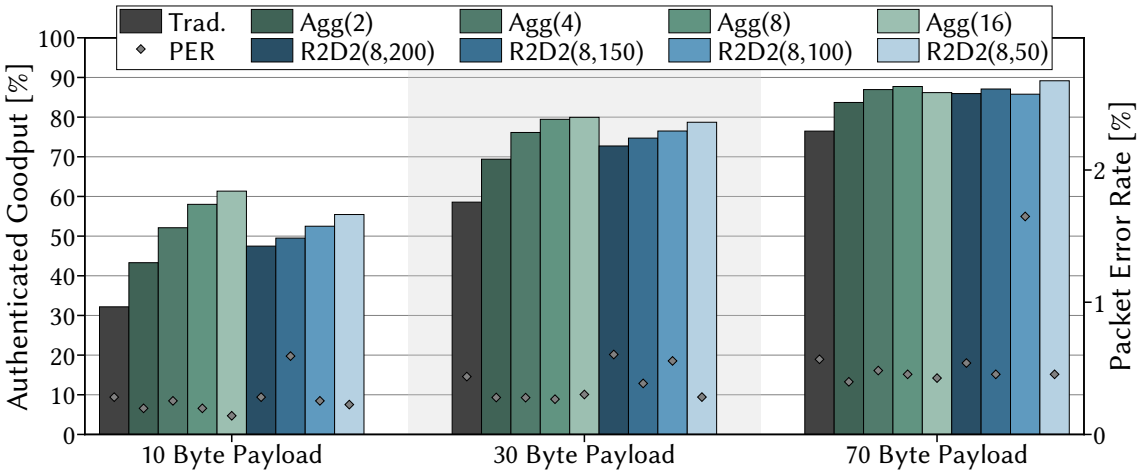


Figure 3.27 When evaluating concrete scenarios, we again see that short payloads benefit most from MAC aggregation with authenticated goodput improvements of up to 90.6 %, 36.5 %, and 16.6 % for 10 byte, 30 byte, and 70 byte payloads, respectively.

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.

3.4.6.3.3 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 3.4.4, our integration of MAC aggregation into DTLS 1.3 offers two processes to deal with this delay. Either any data not yet authenticated is buffered by the DTLS layer and forwarded to the higher communication layer once authenticity is guaranteed. Alternatively, progressive authentication can be employed to optimistically push data to the higher layer and alert in the unlikely case that a falsified MAC is detected retroactively. For both approaches, it is important to understand *how* MAC aggregation delays authentication.

Therefore, we visualize this delay in Figure 3.28 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

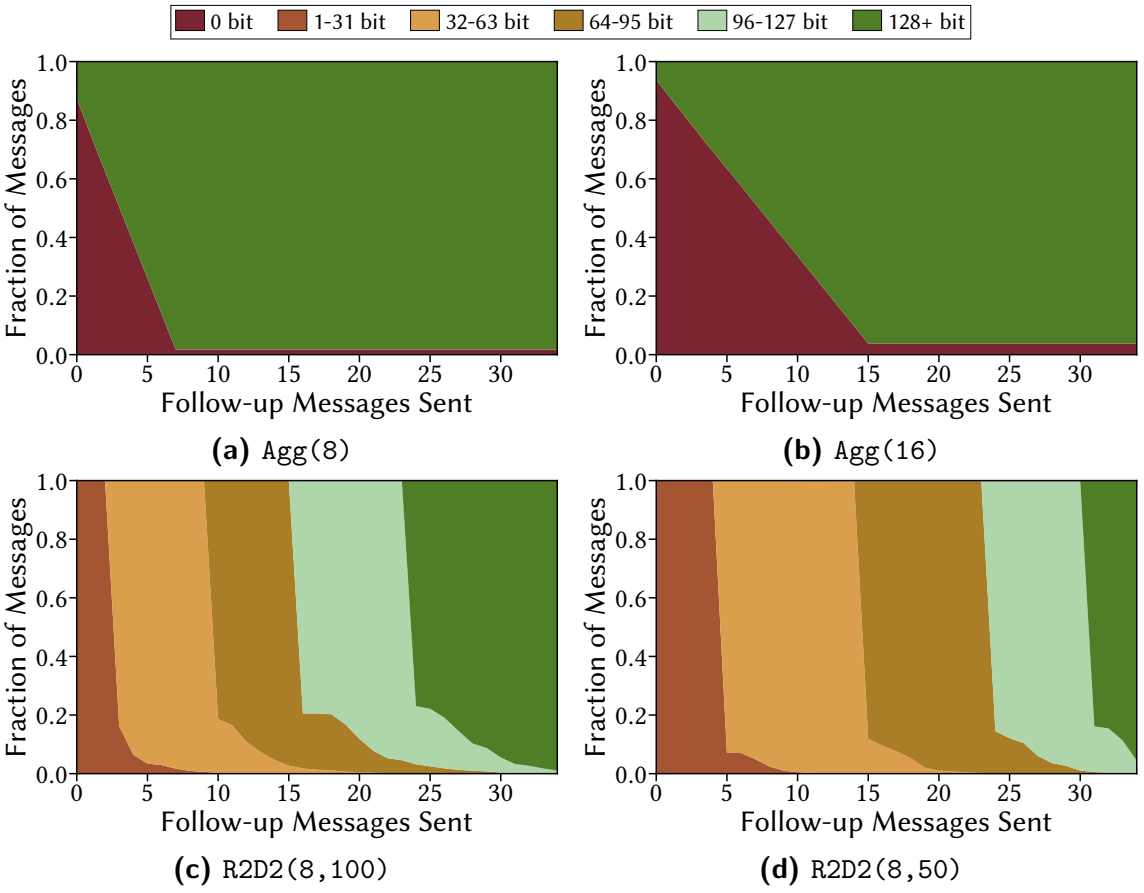


Figure 3.28 Agg(·) provides quicker full authenticity, but some data has to wait long or never receives any guarantees. Meanwhile, R2D2(·) provides progressive authenticity improvements.

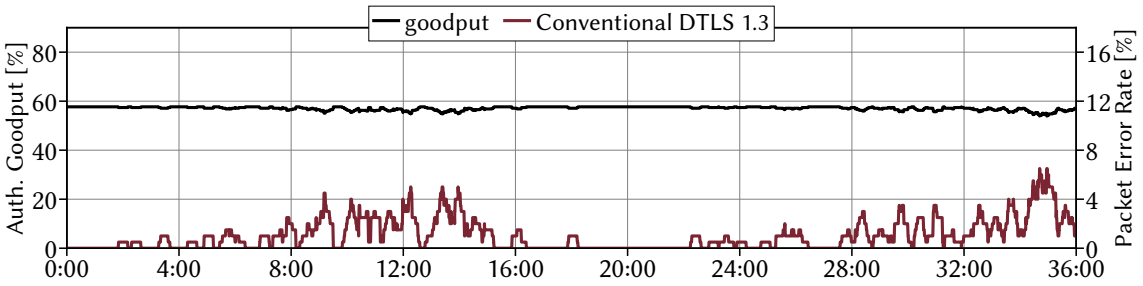
messages have been transmitted with Agg(8), 73.9 % of messages have achieved a satisfactory security level of 128 bit or higher (●) and 26.1 % of messages completely lack any authenticity (●), as can be seen in Figure 3.28a. We see that Agg(·) has lower authentication delays than R2D2(·) and that either 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.

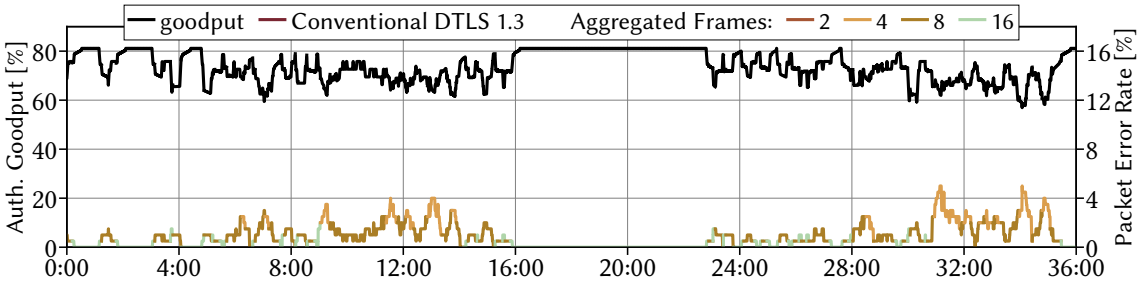
Overall, MAC aggregation leads to authentication delays that can be managed in different ways. These delays differ vastly across aggregation schemes, so an educated choice is advised for a given deployment.

3.4.6.4 Dynamic Parameter Selection

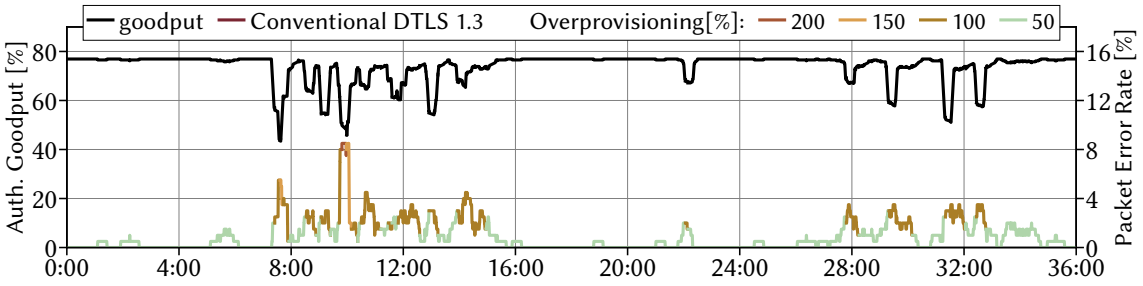
In Section 3.4.5, we presented a mechanism to dynamically adapt the parameters of the MAC aggregation scheme to current channel conditions. Now, we investigate



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



(b) Dynamic MAC aggregtion based on Agg(·) achieves an authenticated goodput of 74.4 %.



(c) Dynamic MAC aggregtion based on R2D2(·) achieves an authenticated goodput of 73.9 %.

Figure 3.29 Dynamic MAC aggregation adapts to varying channels and thus yields significant goodput improvements for short DTLS 1.3 transmissions.

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 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 with a 30 byte of DTLS 1.3 payload.

The results of our evaluation are shown in Figure 3.29. In Figure 3.29a, 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 bottom dark red line shows the running average PER on the second y-axis. Overall, we observe that the PER is low unless the robot is further away from the starting

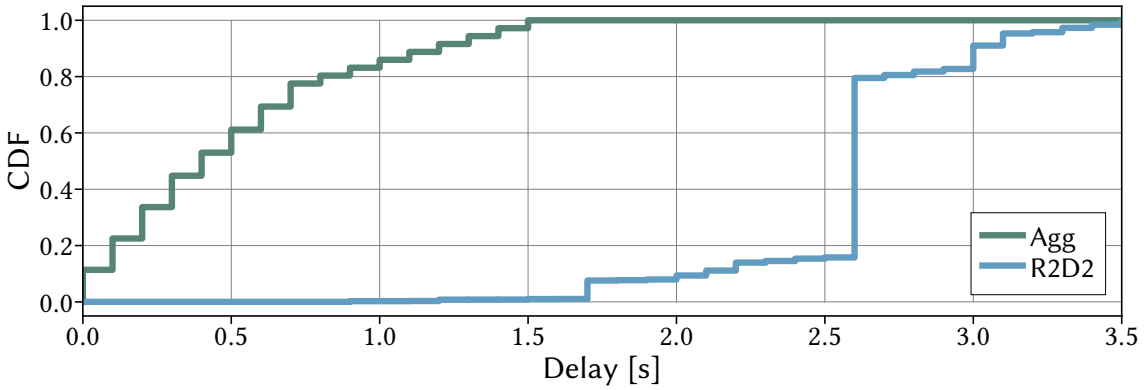


Figure 3.30 `Agg(·)` leads to fast verification but a higher variance in delays than `R2D2(·)`.

point between 7 min and 15 min, and between 28 min and 36 min. Over time, the achieved authenticated goodput remains relatively stable at slightly below 60 %. In total, the conventional DTLS 1.3 connection achieves a goodput of 57 %.

Figure 3.29b now shows the performance of `Agg(·)` with dynamically adjusted parameters. Here, the bottom line not only shows the running average PER on the second y-axis, but also the current MAC aggregation parameters through its color. We observe regular parameter updates while the robot is in motion, selecting more conservative parameters as the channel worsens. Thereby, the authenticated goodput surpasses 80 % on a good channel. Still, 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 3.29c. In contrast to `Agg(·)`, `R2D2(·)` can keep a high authenticated goodput of close to 80 % even when PERs reach 2 %. On the flip side, `R2D2(·)` is then more heavily impacted when PERs spike further. During the spike after 10 minutes, we even see that the channel switches to conventional DTLS 1.3 authentication for a short while. Despite these differences to `Agg(·)`, `R2D2(·)` achieves a similar overall goodput of 73.9 %.

Figure 3.30 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.

3.4.7 Summary

We show that MAC aggregation can indeed 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.

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.

3.5 Conclusion

In this chapter, we tackle the critical challenge of reducing the bandwidth overhead associated with message authentication in resource-constrained and lossy communication environments. We first look at ProMACs, which were recently proposed to reduce this overhead through progressive authentication, *i.e.*, providing initially reduced authenticity guarantees that are then improved over time. However, we uncover a critical vulnerability in these schemes, enabling the proposed sandwich attack to remove all authenticity from targeted messages. We address this vulnerability through Randomized and Resilient Dependency Distributions (R2D2), which builds the foundation for the SP-MAC scheme. SP-MAC is a novel ProMAC scheme that protects against the effects of random or targeted packet loss, while being efficiently implementable on constrained hardware.

Afterwards, we take a step back to study whether the general concept of MAC aggregation can be applied to lossy channels. We show, first through simulations and later in a physical testbed, that MAC aggregation yields strong goodput improvements even on such lossy channels. Our results indicate that, in general, MAC aggregation is particularly effective in scenarios with PERs below 10 % and for payloads shorter than a few hundred bytes. For the measurements in a physical testbed, we integrated MAC aggregation into DTLS 1.3, including a dynamic parameterization to adapt to varying channel qualities.

In the next step, these schemes must be standardized and tested in large-scale deployments to understand their potential fully. The benefits for other protocols, such as *e.g.*, LoRaWAN and Sigfox, should be explored in this context. Overall, MAC aggregation shows strong potential to save valuable bandwidth, and thus energy, in a world with more and more wireless communication nodes.

4

“All we have to decide is what to do with the time that is given us.”

Gandalf, *The Fellowship of the Ring*, 1954

Low Latency Message Authentication for Constrained Environments

Previously, we saw how progressive authentication and MAC aggregation reduce the size of authentication tags to accommodate potential bandwidth and Protocol Data Unit (PDU) size constraints in the Industrial IoT (IIoT). We have, however, not yet covered how message authentication is integrated into legacy communication protocols. While secure protocols for embedded devices, such as DTLS 1.3 [202] or OPC UA [8], have been standardized, upgrading to these protocols is often not feasible for established IIoT deployments [220]. And even when it is possible, encapsulating data in full TLS layers consumes valuable computational resources and bandwidth. Moreover, such an additional protocol layer breaks compatibility with non-updated end devices and middleboxes. Instead, legacy industrial networks can be integrated with retrofittable security mechanisms. Such an integration should remain compliant with existing protocols and consider potential computational constraints. In Section 4.1, we will explore how message authentication can be retrofitted into existing IIoT scenarios with the *Re*trofittable *P*rot*e*ction *L*ibrary (RePeL). RePeL is a flexible library to embed authentication data into unused protocol fields. IIoT devices can thus retrofit a security layer, or offload this step to Bump-in-the-Wire (BitW) devices to save resources, without interfering with existing networking stacks.

However, even with an efficient integration of message authentication, the cryptographic processing itself places a significant burden on time-critical communication. We address this issue in Section 4.2 by proposing BP-MAC, a fast and memory-efficient approach for computing authentication tags based on the well-established Carter-Wegman construction (*cf.* Section 2.4.2). Our key idea is to offload resource-intensive computations to idle phases and thus save valuable time in latency-critical phases, *i.e.*, when time-critical information needs to be communicated. To achieve this offloading, BP-MAC leverages a universal hash function designed for the bitwise preprocessing of integrity protection to later only require a few XOR operations

during the latency-critical phase. Our evaluation on embedded hardware shows that BP-MAC outperforms the state-of-the-art in terms of latency, processing, and memory overhead, notably for small messages, as required to protect IIoT scenarios with stringent security and latency requirements.

4.1 Retrofitting Security in Industrial Protocols

For established IIoT deployment, upgrading to secure communication protocols such as DTLS 1.3 and OPC UA is often impossible due to necessary hardware changes, availability requirements, and the associated cost and risks of breaking a running system [61, 62, 220]. Hence, research focuses on the pressing issue of retrofitting security measures into legacy protocols, with integrity protection being the most important mechanism for Industrial Control System (ICS) security (*cf.* Section 1.2.1).

4.1.1 Related Work

Proposed security retrofitting solutions can be divided into two categories: Extensions to legacy industrial protocols [9,26,32,50,84,216] and the design of additional devices (so-called Bump-in-the-Wire (BitW) devices) that create protected tunnels between themselves and forward legacy communication to existing hardware [18,27,72,208,238, 262]. We discuss the current state-of-the-art with respect to both of these categories in the following.

Considering updates to existing protocols, a first set of proposals simply encapsulate protocols such as ModbusTCP [9] or EtherNet/IP [26] in an additional TLS layer. This encapsulation does, however, introduce significant bandwidth and computational overhead and requires the implementation and support of an entirely new protocol layer. Alternatives propose the integration of authentication tags into existing packets to remain protocol-compliant or reduce the need for additional bandwidth. Here, protocol compliance can be achieved by using the least-significant bits of data values or delays between messages to embed authentication data, which, however, limits security guarantees and restricts which messages can be authenticated [32]. Moreover, the transmission of additional data fields in the Ethernet/IP protocol [50], opportunistically embedding tags into free message space [216] or combining error detection and integrity protection by overwriting the CRC checksum in CAN bus messages [84] have been proposed. Similarly, the IEC 62351-6 [10] standard proposes to embed signatures into reserved fields of the GOOSE protocol.

Most security retrofitting solutions do, however, build on BitW designs to introduce more intrusive changes to the protocol, while still reducing the overhead compared to a full-grown TLS stack. YASIR [238] appends an authentication tag to each packet, while a dedicated verifier module forwards the packet at a reduced rate such that it can corrupt the last byte of the transmission if a packet's integrity could not be verified. Other proposals encapsulate traffic between CAN network segments [27] or append authentication data and change logs (*e.g.*, after protocol translation) to

messages [72]. Moreover, the transmission of dedicated packets containing signatures over recent communications has been investigated with the benefit of not interfering with existing traffic, but at the cost of increased bandwidth needs and delayed attack detection [18]. MARMAC [208], on the other hand, appends multiple tags to NMEA0183 packets to retrofit sender authentication to broadcast communication at the cost of significant bandwidth and computation overhead. Finally, the recently proposed GuardBox [267] is a BitW device that encrypts and authenticates GOOSE messages while protecting keying material within trusted execution environments.

These security retrofit solutions for industrial networks are diverse but mostly focus on individual protocols. Thus, the current state-of-the-art puts little focus on generalizability across the wide range of industrial protocols and variants deployed in the real world. Moreover, how the proposed retrofittable security schemes would be deployed without interfering with ongoing operations is hardly discussed.

4.1.2 Requirements towards a Retrofittable Security Solution

Related work offers legacy-compliant security solutions tailored to specific scenarios and protocols, disregarding the wide variety of industrial environments and the difficulties in deploying proposed solutions. With our work, we want to provide a solution that can be easily adopted to a variety of different protocols, thus not only addressing security issues in common protocols such as ModbusTCP, but also making security much more achievable for more exotic and less-used protocols. To achieve this goal, a *deployable* security retrofit solution must fulfill three main requirements.

Legacy Compliance. To protect legacy devices, authentication data should not interfere with the original communication. Thus, no originally transferred information should be lost, and no additional information (*e.g.*, authentication tags) should interfere with the legacy protocol implementation. Specifically, the receiver of a protected packet should be able to recover the originally sent packet in its entirety regardless of any retrofitting. Moreover, the packet should closely follow the protocol standard while being transmitted, such that potential middleboxes, *e.g.*, rule-based IDSs such as Snort, can be easily adapted to ignore the presence of authentication data. An added benefit of this requirement is stealthy authenticationm, which an intruder may not recognize and then uncover themselves when trying to manipulate allegedly unprotected traffic.

Adaptable to Concrete Industrial Environments. To be widely applicable, a security retrofit solution must be adaptable to various industrial scenarios. On the one hand, this implies that it has to be easily adaptable to new protocols and conditions, depending on how authentication data can be embedded. On the other hand, constraints in terms of available computation resources, tolerable latencies, and approved cryptographic algorithms dictate which authentication scheme can be used. Furthermore, the required security level can vary between applications. While at least 128-bit security is always desirable, some industrial applications with high-frequency data exchanges can be sufficiently protected by as little as 32 bit of authentication data [172]. Hence, the variety of protocols and constraints in industrial networks demands a flexible and adaptable solution for the retrofitting of data authentication.

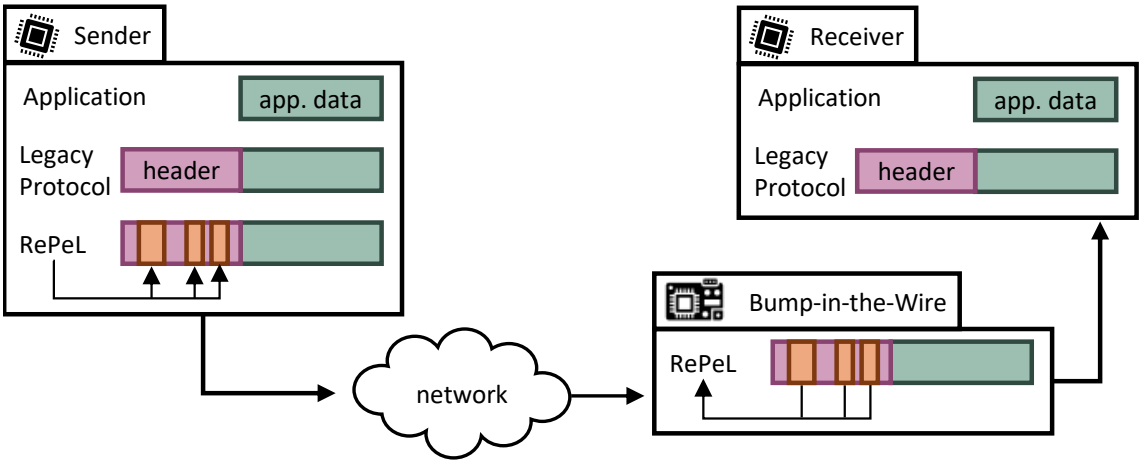


Figure 4.1 RePeL can be deployed natively on devices, embedding or authenticating legacy protocol communication as they exit or enter the networking stack. Alternatively, RePeL can be deployed as a BitW to segment networks or protect devices that cannot be updated.

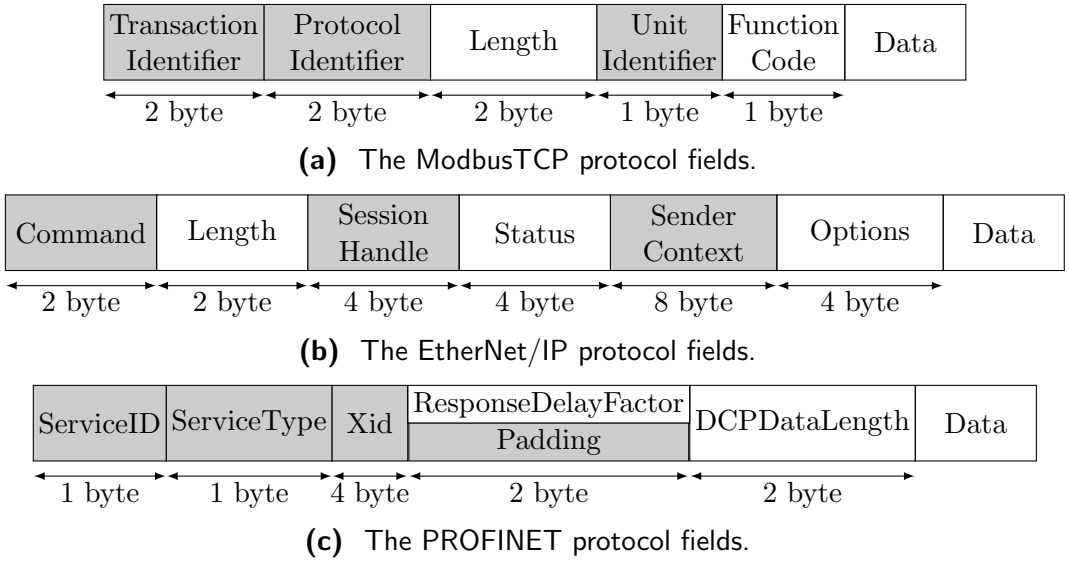
Ease of (Incremental) Deployment. Even if security solutions for concrete industrial scenarios exist, limited updateability of devices and resource limitations prevent adoption in many cases. Hence, security retrofit solutions must be designed to cope with limited resources and be deployable directly on devices for increased security and as a BitW solution for better deployability. The former allows the incremental integration of end-to-end security, while the latter can act as a gateway between already protected and unprotected network segments, and between legacy devices with firmware that cannot be updated. For reducing the risks of interruptions and minimizing downtime during an ICSs operation, providing an incremental upgrade path is inevitable.

4.1.3 RePeL: The Retrofittable Protection Library

To achieve these requirements, we design and implement the *Re*trofittable *Prot*e*c*tion *L*ibrary (RePeL)[1]. RePeL offers a flexible security solution to protect a wide variety of network protocols and deployments with a common solution with minimal configuration. These properties are achieved by integrating authentication data into unused protocol fields of industrial communication protocols, as has been done previously to *e.g.*, introduce covert channels into traditional Internet communication [266].

The high-level approach of RePeL is illustrated in Figure 4.1. The sender and receiver operate identically to a legacy network when RePeL is deployed. Transmitted network traffic is then intercepted either by a lower layer of the communication devices, respectively shortly after transmission or before reception by a dedicated BitW device. When intercepted, the legacy protocol header is enriched with authentication data. The packet is thus integrity protected until this authentication data is subsequently removed by the receiving RePeL instance that recovers the original packet.

[1] `https://github.com/fkie-cad/RePeL`



(a) The ModbusTCP protocol fields.

(b) The EtherNet/IP protocol fields.

(c) The PROFINET protocol fields.

Figure 4.2 The three analyzed industrial protocols (ModbusTCP, EtherNet/IP, and PROFINET) each exhibit potential to embed authentication data (highlighted in gray).

4.1.3.1 Threat Model

For the design of RePeL, we consider an attack that aims to manipulate messages sent over a communication channel in industrial networks. Our attacker already gained access to the network but has no control over the two entities involved in the targeted communication as defined in the Dolev-Yao threat model [67]. Accordingly, the attacker can arbitrarily read, alter, reroute, inject, and drop packets. Within our threat model, RePeL should prevent the attacker from undetectably changing messages. Denial of Service (DoS) attacks, fingerprinting, and side-channel attacks are out of scope for the design of RePeL as these attacks affect any security protocol.

4.1.3.2 Analysis of Repurposable Protocol Fields

To assess the potential of RePeL, we analyze the feasibility of directly integrating authentication data into unused fields in protocol headers and the subsequent recovery of the original packet without information loss. In particular, we look at three popular ICS protocols, namely EtherNet/IP, PROFINET, and ModbusTCP [104]. Throughout their analysis, we observe common patterns in protocol header designs that RePeL can take advantage of. To this end, Figure 4.2 highlights suitable fields in the analyzed protocols headers (in grey), revealing the presence of unused bits across many different header fields. Furthermore, we provide a concrete description of how and when these header fields can be leveraged for integrity protection in Table 4.1. In the following, we discuss the different patterns in more detail.

4.1.3.2.1 Message Identifiers

Message identifiers are a concept for matching requests and responses used by all three analyzed protocols. These fields are typically 16 to 32 bits long and can be

freely selected by the sender of a request. The receiver of a request then echoes the field, such that the requester can match the response to the correct request. This behavior already allows the fully protocol-conform unidirectional transfer of authentication data by using a truncated authentication tag as message identifier. This procedure can also be adapted to enable bidirectional message authentication: by using only the, *e.g.*, 16, most significant bits of the identifier for request-response matching, the rest of the field can carry authentication data. Then, the responder would echo the first part of the identifier and compute new authentication data for the response that fills the second part of the identifier. As fewer identifiers than available are encodable concurrently (e.g., ModbusTCP has 65536 identifiers while only allowing 16 unanswered requests), no impact on the operation of devices is expected, especially as these changes do not prevent RePeL from exposing the original behavior of message identifiers to the application layer.

4.1.3.2.2 Unused Fields for Legacy Reasons

For backward compatibility, fields such as the Unit Identifier in ModbusTCP exist to differentiate multiple serial Modbus devices located behind one ModbusTCP gateway. However, most network deployments nowadays do not contain such devices. Thus, the corresponding fields remain unused, and no harm is done by redefining the interpretation of the contained data. Such fields with hardly any relevance today can thus be reused in many cases. However, vendor-specific requirements for static values in these fields might have to be considered before packets are forwarded to legacy devices.

4.1.3.2.3 Reserved Bits

The analyzed protocol headers contain a sizable number of reserved bits for future extensions. Ideally, one of these bits can be used to indicate that the rest of them are used for authentication data. Then, future extensibility of the protocols remains possible, while the currently unused bits can be put to good use by carrying authentication data. As these bits are expected to be set to a static value (0 in most cases), RePeL can recover the original packets after integrity verification and before forwarding packets to legacy applications.

4.1.3.2.4 Padding Bits

A final common component of communication protocols are padding bits to *e.g.*, adjust the length of different header variants. PROFINET, for example, uses padding to ensure that broadcast and unicast messages have the same length. Such adjustments can ensure faster parsing or prevent information leakage if traffic is encrypted. These padding bits can be redefined to carry authentication data. While some protocols ask for a static value, no information is lost if the content of such a field is replaced with authentication data, as the recovery of the original packet remains possible.

Summarizing our findings, we conclude that unused bits are present in each of the analyzed protocol headers, as detailed in Table 4.1. The most space to embed

Protocol	Header Field	Size	Reusable	Usage Description
ModbusTCP 8 byte header ↓ up to 36 free bits	Transaction Id	16 bit	12 bit	As the standard limits the maximum number of concurrent requests to 16, four bits are sufficient for request/response matching.
	Protocol Id	16 bit	16 bit	The standard suggests zeroing this field, such that it carries no information.
	Unit Id	8 bit	0-8 bit	This field is used to identify serial Modbus devices in a ModbusTCP network, which rarely exist. The field carries no additional information beyond this identification.
EtherNet/IP 24 byte header ↓ up to 63 free bits	Command	16 bit	7 bit	The first byte is comprised of reserved bits, of which the first one can mark that the remaining seven bits can be used for authentication data.
	Session Handle	32 bit	0-24 bit	Used to identify a communication session. Yet, even the decently sizes SWAT testbed [86] requires fewer than 200 session ids over an one-hour period.
	Sender Context	64 bit	0-32 bit	Matches requests to responses. Like other protocols, the full range of identifiers is rarely needed; Thus, some bits can be used for bidirectional authentication data.
PROFINET 10 byte header ↓ up to 40 free bits	Service ID	8 bit	3 bit	The first nibble is comprised of reserved bits, of which the first bit can mark that the remaining three bits can be used for authentication data.
	Service Type	8 bit	5 bit	Six bits are reserved for future extensions and could be used to integrate authentication data. Again, one bit could notify about the embedded authentication data.
	Xid	32 bit	0-16 bit	Matches requests and responses. Like other protocols, the full range of identifiers is rarely needed, such that some bits can be used for bidirectional authentication data.
	Padding	16 bit	16 bit	Unused 2-bytes field to extend the length of unicast frames that can be defined to carry authentication data.

Table 4.1 Our analysis of three industrial protocols identifies several fields that can be (partially) used to embed more than 32 bit of authentication data. For each field, we indicate the number of realistically reusable bits and how they could be used.

authentication data can be found in the EtherNet/IP header, with up to 63 bits. The other protocols also offer up to 36 bits for ModbusTCP and 40 bits for PROFINET. As we observe similarities across the analyzed protocols, we expect that many further industrial protocols expose similar behavior. However, we also observe that this space is fragmented and that, depending on the specific network, updateability,

and required level of protocol conformance, a different fraction of those bits can be repurposed. Moreover, the previously outlined requirements towards latency, approved cryptographic algorithms, and requested security level demand a flexible and adaptable use of the limited available space to offer protection for industrial protocols. We specifically argue for the use of available space in industrial protocol headers to carry authentication data. However, related work also proposes embedding authentication data into *e.g.*, the least significant bits of noisy measurement data, as the impact of these changes is negligible to industrial processes running in the background [32]. RePeL adopts such approaches for maximizing the available space for authentication data. We conclude that industrial protocols offer sufficient unused space in various application scenarios that can be used to protect the integrity of industrial communications.

4.1.4 Design of RePeL

To take advantage of unused space in protocols headers, we design the *Re*trofittable *P*rot*e*ction *L*ibrary (RePeL). RePeL embeds authentication into headers before transmission and verifies the integrity of packets upon reception. We first present RePeL's high-level modular design, before diving into the specifics of each module.

4.1.4.1 High-level Design

RePeL protects legacy traffic between a RePeL sender and one or multiple receiving endpoints by embedding authentication data into the packets. This insertion is later verified and removed upon packet reception by the receiving endpoints. To see widespread applicability, RePeL needs to be adaptable to the local requirements of different ICS environments. Most importantly, these requirements can include the support for specific protocols, the latency (translating to a limited processing time), legal requirements w.r.t. allowed security schemes, and the tolerance for packet loss.

To achieve this kind of customizability within a single framework, RePeL is composed of three modules: The *Parser module*, the *MAC module*, and the *Nonce module*. The interaction of the different modules is shown in Figure 4.3. To protect a packet, it is first parsed to extract the location of bits that can be used for authentication data. Then, the packet is brought into a deterministic state, which can be recovered even after the authentication tag is embedded. One this way, we can generate and verify the authentication tag over the exact same data. Finally, the MAC and the Nonce modules can jointly compute an authentication tag, which is subsequently embedded as authentication data.

At the receiving end, RePeL verifies the previously embedded authentication tag. Therefore, the embedded bits are first extracted from the packet before it is brought back into the previously discussed deterministic state. Then, a new authentication tag is computed over the received packet and compared to the extracted tag. If both of them match, the packet's integrity is verified. Otherwise, an alarm is raised, and the packet is not forwarded for further processing. In the following, the three RePeL modules are presented in detail after discussing key management within RePeL.

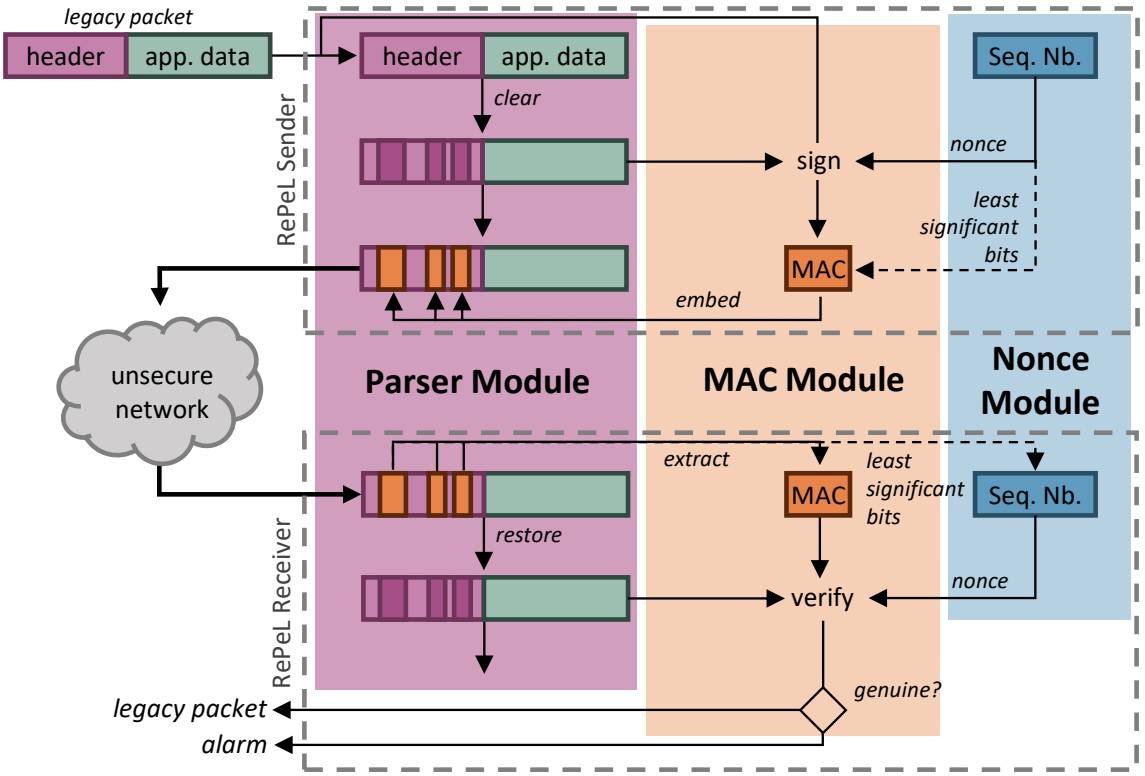


Figure 4.3 RePeL is composed of three modules to embed and verify authentication tags. These tags are computed by the sender and receiver over a common packet state that each packet is first brought into. If the validation of a received packets succeeds, it is forwarded to the next layer and discarded otherwise.

4.1.4.2 Key Establishment and Resynchronization

Before diving into the details of how RePeL authenticates messages, we first want to discuss key management. ICSs are characterized by being operated by a single organization, and often networks remain mostly static, *i.e.*, no new communication entities join a network. Therefore, secrets shared only by two entities that have to communicate can be established during the deployment of RePeL by the operator. From this shared secret, authentication keys can be derived. This key derival can be triggered automatically, *i.e.*, on first communication of when the nonce counter is about to overflow, or by special messages communicated only between the RePeL instances (thus no breaking legacy compliance with the application layer). Alternatively, well-established key agreement protocols between the RePeL instances can be employed to establish the necessary keys.

If RePeL were deployed in a more dynamic setting, communicating instances might not share a secret when first wanting to communicate. Here, traditional key establishment as specified in *e.g.*, TLS 1.3 [200] could be used, either in plaintext or over a prolonged sequence of covert messages. Most importantly, this key exchange can be conducted during production downtime, when latency and bandwidth limitations are significantly more relaxed.

4.1.4.3 Highly Adaptable Protocol Parsing Module

The main module of RePeL takes care of parsing packets in an efficient and modular way. This parsing includes the identification of free space for authentication tag inclusion, embedding and extracting the tags at the sender and receiver, respectively, and the restoration of packets to their pre-embedding state for tag computation and further processing.

RePeL's parser executes its duties with three passes over a packet for efficient parsing. In the first step, the number of available bits for authentication tags is *extracted*. The second step *clears* the packet of any potential inconsistencies to a form that can also be reproduced by the receiver. This step, for example, includes mapping long message identifiers to their shortened form, which allows the inclusion of authentication data. Finally, the *embedding* or *extracting* of the authentication tag is done in a final pass over a packet.

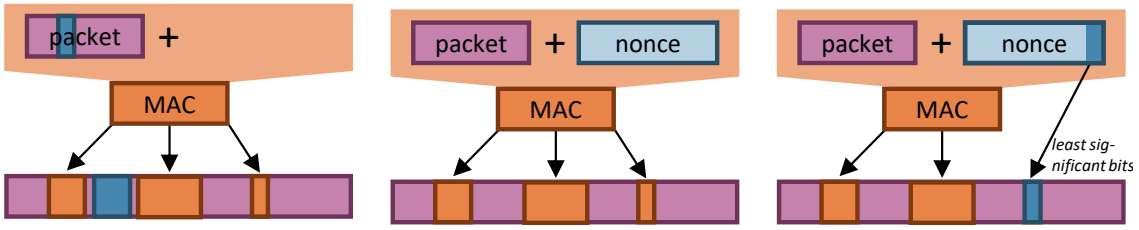
To remain adaptable, RePeL provides custom functions to *peek* into, *skip* over, or *manipulate* individual bits of a packet header. This structure enables fast parsing, while remaining highly adaptable: A single line in a protocol's definition can be changed to include or exclude *e.g.*, the protocol identifier of ModbusTCP as a field that can be retrofitted with authentication data. This flexibility allows a fast adaptation to the requirements of vastly different ICS deployments.

4.1.4.4 MAC Module

RePeL's second module computes the authentication tag that is subsequently embedded into packet headers. These authentication tags are computed by a Message Authentication Code (MAC) scheme, *i.e.*, a combination of a *sign* and *verify* cryptographic algorithm. The sign algorithm computes the tag based on the data to be authenticated, a secret key shared between sender and receiver, and an optional nonce for replay protection. The verify algorithm compares the received tag to a locally computed version based on the same inputs. If both tags match, the packet is genuine and rejected otherwise.

To function properly, both algorithms need the same data as input, which is why a packet is first cleared/restored (*i.e.*, to a state that sender and receiver can reproduce) before an authentication tag is computed over this data. Note that using the restored packet, instead of defining individual blocks of the message that should be skipped during authentication, saves valuable memory and computation resources on constrained legacy devices, as all computations can be done in-place.

The MAC module further contributes to RePeL's adaptability, as the optimal message authentication code (MAC) scheme selected may depend on different requirements and available resources. As such, the common HMAC-SHA256 algorithm might be supported by hardware acceleration or be mandated by regulation. However, if stronger security guarantees are required with limited available space, aggregated [115] and progressive MAC schemes [20, 246] may be advantageous. Similarly, latency-critical applications might require fast general schemes such as UMAC [127] or even faster schemes optimized for short messages such as BP-MAC [250].



(a) For protocols including a nonce, replay protection is naturally provided.

(b) For reliable protocols (*e.g.*, TCP), an implicit counter acts as nonce.

(c) To handle packet loss, the least significant bits of a counter can be transmitted.

Figure 4.4 Depending on the deployment scenario, efficient replay protection is provided with three different mechanisms.

4.1.4.5 Integrating Nonces for Replay Protection

The parser and MAC modules allow RePeL to efficiently integrate authentication tags into messages. However, MAC schemes generally do not provide replay protection, which is crucial to prevent an attacker from intercepting and later injecting an already authenticated message at an (in)convenient time. To mitigate such attacks, MAC schemes can include nonces as unique additional input to tag computations that prevent a replayed message from being considered benign.

However, appending a nonce to each message requires space that is often not available. Hence, RePeL offers three methods for replay protection that are shown in Figure 4.4, which can be selected depending on a scenario's needs.

Figure 4.4a highlights the trivial method, where replay protection is natively provided at the application layer protocol. This approach is applicable if the industrial protocol includes a nonce of some form, which is automatically checked for uniqueness by the application layer, *e.g.*, an increasing sequence number. Then, the authentication tag can be computed over the message already including a nonce.

Most industrial protocols do, however, not include a nonce in their headers, such that RePeL needs to provide replay protection. Here, RePeL uses a zero-initialized counter as nonce that counts the number of sent packets and includes this number during tag computations and verifications. As shown in Figure 4.4b, this nonce does not necessarily need to be embedded into a packet. Instead, if a reliable communication channel with no data loss (*e.g.*, TCP) is used, the nonce can be tracked by both communication partners without active communication. Thus, no valuable space is consumed, while the MAC module still has access to the same unique nonce at the sender's and receiver's end.

To accommodate for packet loss on unreliable channels, *e.g.*, UDP, without transmitting the entire nonce with each packet and thus occupying space for authentication data, RePeL relies on a similar mechanism as used by MiniSec [154] and in the proposed DTLS 1.3 standard [202]. Instead of transmitting the entire nonce (a zero-initialized counter) with each packet, only its, *e.g.*, four least-significant bits are embedded into the packet header following the authentication tag as shown in Figure 4.4c. As our nonces are sequential, the transmitted bits allow us to detect dropped packets and recover the full sequence number of a sent packet. If, *e.g.*, three

packets are lost, the received partial nonce is off by three from the expected sequence number, and the receiver knows that the actual count can only be higher than its internal state. Hence, the receiver can add three to its counter and use that value to verify the authentication tag. This method computes the correct nonce as long as error bursts are not longer than what is trackable by the embedded bits. Thus, before deployment, the number of bits reserved for nonce transmission should be carefully chosen to optimize the amount of available authentication data and achieve a low desynchronization probability. If an authentication tag verification fails, this must be communicated to the sending RePeL instance. Then, the RePeL keying can be reset (*i.e.*, new keys are derived and the nonce reset to zero), and further messages are no longer rejected due to synchronization issues following a long burst of errors.

Overall, RePeL provides flexibility and adaptability in terms of protocol parsing, authentication scheme selection, and the mechanism used for replay protection. Thus, RePeL is not another retrofitting attempt tailored to a specific deployment but a customizable solution to optimally protect a wide variety of industrial scenarios where other security solutions do not meet performance or legal requirements or simply do not exist.

4.1.5 Performance Analysis of RePeL

Beyond the technical feasibility of embedding authentication data in legacy packet headers, the achievable performance on embedded hardware is also critical to understand RePeL's applicability. Legacy devices, but also newer industrial IoT devices, typically offer reduced processing power that now need to execute expensive cryptographic operations [220]. To show that the overhead of RePeL is well-manageable even for severely resource-constrained devices, we consider two devices representative of low-power embedded hardware: the Zolertia Z1 (MSP430 @ 16 MHz, 8 kB RAM) and the Zolertia RE-Mote (ARM Cortex-M3 @ 32MHz, 16 kB RAM).

For our evaluation, we first design a RePeL instance for the ModbusTCP protocol. We then analyze the latency overhead introduced by RePeL and how it is influenced by packet lengths. Furthermore, we compare how native security deployments fare against BitW approaches that introduce specific hardware to handle the authentication of legacy traffic.

4.1.5.1 A RePeL Instance to Protect ModbusTCP Traffic

To demonstrate the utility and usability of RePeL, we configure a ModbusTCP parser, leveraging the highly adaptable design of the parser module, and taking advantage of the three fields proposed in Table 4.1 for ModbusTCP. Here, we discuss the sender side of the code in more detail while noting that the receiver uses similar variants of these functions.

At first, the number of embeddable bits is extracted, which may require a first pass over the header or is fixed for a given RePeL instantiation. In our case, 32 bits are available, which is sufficient to achieve adequate integrity protection [172], which

is thus propagated to the MAC module. Afterward, the header is cleared, *i.e.*, the reused fields are set to a deterministic value that can be recovered by the receiver, as shown in Algorithm 1. Here, RePeL offers predefined functions that allow quick restoration of most fields, *e.g.*, the clearing of the protocol identifier field to zero. These are used to, *e.g.*, learn the packets' transaction identifier and replace it with a shortened one. We store this mapping to later replace the shortened identifier with the original one once a response packet arrives, such that our modification remains transparent to the application layer protocol. Finally, our clear function for ModbusTCP skips over the length fields and enforces the unit identifier field to be `0xFF`, the default value for ModbusTCP.

Algorithm 1: Clear Functionality for ModbusTCP

Input: packet *pkt*

```
tid = peek(&pkt, 16)                  ▷ extract the Transaction Identifier
copy(&pkt, new_tid, 4)                                ▷ embed new 4 bit id
push(&pkt, 0, 12);                                    ▷ clear remaining bits
store_mapping(tid, new_tid);
push(&pkt, 0, 16);              ▷ set Protocol Identifier to expected value
skip(&pkt, 16);                                       ▷ skip over Length field
copy(&pkt, 0xff, 8);              ▷ set Unit Identifier to expected value
```

Algorithm 2 then shows how the tag computed over the restored packet by the MAC module is embedded into the ModbusTCP packet header. Unmodified (partial) fields, such as the most significant bits of the transaction identifier, are skipped, while the rest of the fields are replaced with the segmented authentication tag.

Algorithm 2: Embed Functionality for ModbusTCP

Input: packet *pkt*, authentication tag *tag*

```
skip(&pkt, 4);                                ▷ skip Transaction Identifier
copy(&pkt, &tag, 12);
copy(&pkt, &tag, 16);                      ▷ overwrite Protocol Identifier
skip(&pkt, 16);                                       ▷ skip over Length field
copy(&pkt, &tag, 8);                     ▷ overwrite Unit Identifier field
```

4.1.5.2 Influence of Packet Length on Delays

To assess RePeL's performance on constrained devices, we first look at the mean processing time of RePeL for native deployments and how it scales with varying payload lengths. Therefore, we authenticate varying-length payloads and embed a 16-byte authentication tag within the header. Having achieved comparable results on the receiver side, we only show the sender side measurements here. The used authentication scheme is HMAC-SHA256. For each length, we authenticate 1000 packets and repeat this measurement 30 times. Figure 4.5 includes the 99 % confidence intervals, which are barely visible. The measurements are repeated for the Zolertia

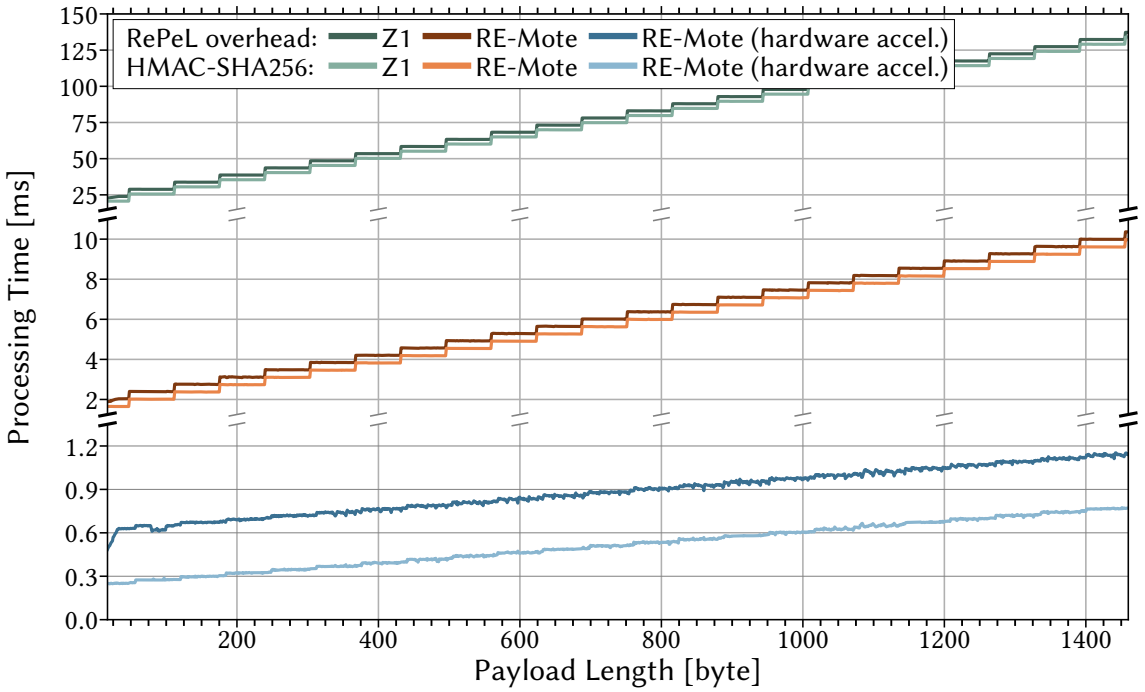


Figure 4.5 Native deployments of RePeL on embedded hardware can have vastly different processing times depending on available resources. This processing overhead is dominated by the computations of HMAC-SHA256 digests.

Z1 and Zolertia RE-Mote. For the RE-Mote, we also repeat the measurements with enabled hardware acceleration for SHA256.

In total, we see significant, but often acceptable [155], overhead ranging between 22.71±0.02 ms and 137,27±0.03 ms on the Zolertia Z1. The observed staircase-like increase in overhead can be explained by the block size of the used cryptographic hash function; if the payload size increases by 64 bytes, an additional block has to be computed, which is reflected by the increased processing overhead. As the processing is dominated by computing the cryptographic hash function, the processing time is nearly constant between these jumps and thus independent of payload lengths.

For the software implementation of RePeL on the Zolertia RE-Mote, we see a similar picture, with the main difference being that processing times lie between 1.88±0.02 ms and 10.36±0.02 ms. Thus, the RE-Mote is over an order of magnitude faster. Still, most processing is required to compute the cryptographic hash function. Using the hardware-accelerated `SHA256` implementation, we see another speedup of nearly an order of magnitude. Here, the overhead introduced by the cryptographic hash function is no longer dominating, and we see a more constant increase in processing time w.r.t. the payload size. Overall, processing overhead is low enough that even constrained industrial devices can handle them [155].

Most importantly, we see that RePeL's overhead is dominated by the processing of HMAC-SHA256. Consequently, the overhead of deploying RePeL on comparatively weak hardware, represented by the Zolertia Z1, can be intolerable for some applications. For such cases, RePeL's modular design offers the use of more suitable MAC schemes, *e.g.*, UMAC [127] or BP-MAC [250], if higher performance is required.

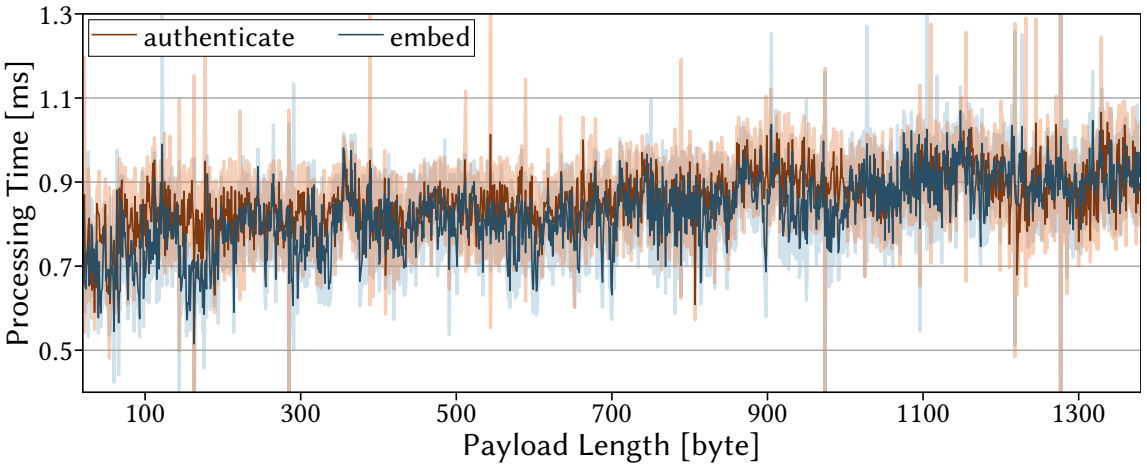


Figure 4.6 RePeL deployed on a NanoPi R2S as a BitW reduces processing times at the expense of increased jitter. The lighter colors indicate 99 % confidence intervals.

4.1.5.3 RePeL as a Bump-in-the-Wire deployment

Many proposals to retrofit security into ICS communication take advantage of a BitW approach to protect the network [18, 27, 72, 208, 238, 262]. Here, special-purpose devices are added to bridge legacy hardware and the network or to segment networks. The devices will then ensure secure communication between themselves. Only the last hop to the legacy hardware remains unprotected. RePeL is the first security retrofitting solution that offers both native as well as BitW deployments. This dual usage possibility enables the incremental deployment of true end-to-end security but also helps us to assess the direct performance comparison between the native and BitW security retrofitting solutions.

To this end, we deployed RePeL on a NanoPi R2S to measure the delays of an additional networking stack introduced by the deployment of an additional device to embed and verify authentication data into packets. We measured this delay for UDP packets with payload lengths ranging again between 20 and 1380 byte. Figure 4.6 shows our results of 30 measurements of packets for each length, including 99 % confidence intervals. We observe that the superior processing power of the NanoPi R2S more than compensates for the overhead introduced by passing the Linux networking stack two additional times, with delays mostly staying below 1 ms. It is thus also hardly surprising that the actual packet length, in contrast to native deployments, has little influence on the overall delay. Meanwhile, we see relatively increased jitter in these measurements. Hence, BitW deployments offer lower latencies at the expense of increased jitter, which can be problematic in ICSs [89, 233].

4.1.6 Summary

The wide variety of legacy protocols in IIoT scenarios and their plentiful flavors call for adaptable mechanisms to retrofit security that protect industrial control systems against modern cyberattacks. However, current solutions often focus on specific protocols and leave wide-scale deployability as an afterthought. To address

these shortcomings, we propose RePeL, a modular and adaptable framework to embed authentication tags into unused protocol header fields, which are widespread in industrial protocols. RePeL's adaptability enables customizing which fields are used depending on the protocol and the concrete deployment. RePeL's modularity furthermore allows to, *e.g.*, choose a dedicated authentication scheme based on the deployment's needs, such as fast processing, low space demands, or regulations. Our prototypical open-source implementation of RePeL shows strong performance both when deployed natively on embedded hardware as well as on BitW solutions ensuring secure communication between RePeL instances.

4.2 BP-MAC: A Bitwise Preprocessable MAC Scheme

Even with an efficient security retrofitting solution, a major latency and processing bottleneck still exists in the computations of the generation and verification of authentication tags. Figure 4.5 demonstrates, for example, how RePeL's processing time is dominated by the computations of the authentication tag. Even with hardware-acceleration for `HMAC-SHA256`, its computation still consumes approximately half of all processing time already for short 1-byte payloads.

Consequently, choosing the right MAC scheme is important to minimize processing delays and overhead, especially on slower embedded hardware. There exists a large selection of different MAC schemes; However, none of them are optimized for short messages that are only a few bytes long. Meanwhile, these are exactly the type of messages mostly encountered in latency-critical IIoT scenarios where the constrained embedded hardware demands efficient processing of cryptographic schemes. To fill this gap, this chapter introduces the *Bitwise Preprocessable MAC* scheme (BP-MAC)[2]. BP-MAC offloads resource-intensive cryptographic computations to idle phases and thus saves valuable time in latency-critical phases, *i.e.*, when new data awaits transmission.

4.2.1 Related Work

For most of the Internet, well-established MAC schemes such a `HMAC-SHA256` provide sufficient efficiency. For use cases that require more efficient options, such as securing resource-constrained devices, optimized computations of established primitives, such as AES or SHA256, are available. Such optimizations include hardware acceleration [263] or the preprocessing of authentication for predictable parts of future messages [101]. However, these approaches still fail to enable secure sub-millisecond communication as many IIoT scenarios demand [82, 155]. Consequently, a range of novel MAC schemes based on new cryptographic primitives has been proposed to reduce processing and latency overheads, such as SipHash [21], TuLP [87], Chaskey [165], or LightMAC [156]. MergeMAC [17], on the other hand, saves valuable time during latency-critical phases by extracting predictable parts of future

[2] `https://github.com/fkie-cad/bpmac`

messages to authenticate them in advance. However, mission-critical IIoT must rely on scrutinized and time-proven cryptographic primitives to ensure the maximum possible security guarantees.

Therefore, competing approaches target efficient Carter-Wegman MAC constructions, whose security is derived from the underlying well-established cryptographic primitives, *e.g.*, AES. To do so, UMAC [127] and VMAC [126] use custom universal hash functions optimized for 32-bit and 64-bit architectures, respectively. Poly1305, on the other hand, evaluates a polynomial with the message as coefficients over the finite field $\mathbb{Z}_{2^{130}-5}$ [34]. Although highly efficient, these schemes are optimized for messages of at least a few hundred bytes [21] and not the usually short messages of the IIoT (*cf.* Section 1.2.1). Likewise, PMAC [38] aims to fragment large messages to parallelize tag computations, which typical resource-constrained devices with single-core processors cannot benefit from.

In reality, IIoT networks are increasingly exposed to potent attackers with adverse goals and thus need to be protected with the security of well-established cryptographic primitives with a proven track record of resisting powerful cryptanalysis. Meanwhile, these scenarios are often latency critical, where as little time as possible can be wasted on cryptographic processing. As most MAC schemes are not optimized for the short messages of only a few bytes usually seen in the scenarios, we expect that significant optimization performance is left unexplored. This motivates our following quest to design a MAC scheme that combines the security provided by established primitives with optimizations targeted at short messages.

4.2.2 Precomputations for Fast Authentication

BP-MAC takes advantage of precomputations since the processing loads in IIoT scenarios are often not evenly distributed over time [17, 101]. Instead, devices are regularly idling until new data has to be transmitted, received, or processed. Hence, the idea behind BP-MAC is to offload computationally expensive operations into those idling phases to minimize computations during latency-critical phases. A naïve approach to fulfill this goal is to precompute and store authentication tags for each possible message. However, precomputing authentication tags for all possible, *e.g.*, short two-byte messages would already consume 1 MB of memory (assuming the recommended tag length of 16 bytes), a number that grows exponentially for longer messages. Hence, this naïve approach is infeasible for resource-constrained devices.

In contrast, BP-MAC is a Carter-Wegman construction, with precomputed tags for individual bits, that are then efficiently combined within the universal hash function when a message needs authentication. As each bit can only represent either 0 or 1, BP-MAC's memory consumption grows linearly with message lengths and amounts to only 512 bytes for *all* possible two-byte long messages. These tags can be precomputed once a new key is exchanged or even provided by a potentially more powerful communication partner. In addition, BP-MAC masks the combined tags with a secret blinding tag to prevent replay attacks and enable their secure aggregation. Such blinding tags can be conveniently precomputed between transmissions since they are independent of the actual message.

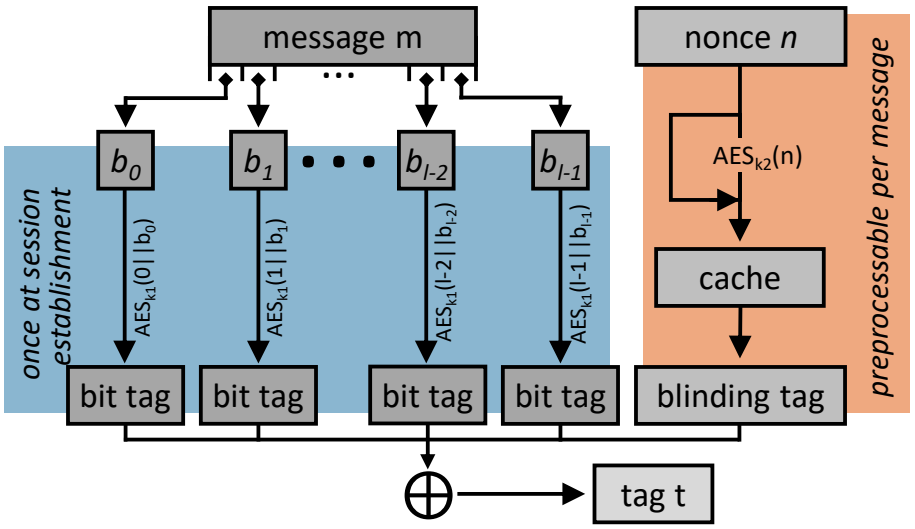


Figure 4.7 BP-MAC achieves high-speed tag computations during the latency-critical phase by only XORing the bit tag depending on each bit value and the blinding tag to generate a secure authentication tag. The blinding tag is independent of the message and can thus be precomputed during idle times. Bit tags are precomputed once for each unique key.

4.2.3 BP-MAC's Design in Detail

BP-MAC achieves fast authentication by only executing the efficient aggregation of precomputed tags during the latency-critical phase. We illustrate the different steps of BP-MAC's design in Figure 4.7 and discuss the details of the different tags as well as how they are computed and verified in the following.

4.2.3.1 Bit Tags

As shown in Figure 4.7, bit tags are intermediate tags computed for each message bit. To ensure BP-MAC's security and prevent collisions, all bit tags must be unique pseudorandom numbers only known to the senders and receivers of messages. BP-MAC computes bit tags by applying a pseudorandom function to the concatenation of the bit index and bit value. Concretely, BP-MAC uses AES-128$_{k1}$ as a pseudorandom function. Here, $k1$ is a secret key shared between both communicating parties due to its increased efficiency over other functions, *e.g.*, HMAC-SHA256. Thus, if the value of the third bit in the second byte is zero (*i.e.*, the overall 10^{th} bit), the corresponding bit tag is AES-128$_{k1}$(`0b1010||0b0`). Overall, there exist $2 \cdot l$-bit tags that have to be precomputed for a message with exactly l bits. However, this only needs to happen once a new key is established and could even be offloaded to a more powerful device if necessary.

4.2.3.2 Blinding Tags

The purpose of blinding tags is twofold. First, they provide replay protection by incorporating a nonce into the final tag. Second, these pseudorandom tags ensure

that no information about the individual bit tags is leaked and are thus crucial for the security of BP-MAC. As nonce, BP-MAC uses a zero-initialized counter that is incremented for each newly computed tag.

Computing the blinding tag from a pseudorandom function and the nonce is analogous to UMAC [127] and relies on AES-128$_{k2}$. For tags with a length between 9 and 16 bytes, we use the leading n bytes of the AES-encrypted nonce, where $k2$ ($\neq k1$) is also a secret key shared between both parties. For smaller tags, we cache the 16-byte long AES output for multiple blinding tags, only invoking AES again if not enough output bytes are available for the next blinding tag. Thus, for tags of 4 bytes, only every fourth nonce has to be encrypted to reduce processing overhead, while we can use the unused bytes of the cached output to blind the remaining tags. As nonces can be predicted, blinding tags can be precomputed and thus do not add to the latency-critical delay.

4.2.3.3 Tag Computation and Verification

After discussing the two precomputation phases generating the bit and blinding tags, we now explain the computation of the authentication tag t_n for the n-th message m_n in a message stream. As shown in Figure 4.7, the final step of tag computation consists of XORing all intermediate tags, shown in the formula for the tag t_n:

$$\mathrm{t_n} = \bigoplus_{i \in |m_n|} \overbrace{\mathrm{AES\text{-}128_{k1}}(i||m_n[i])}^{\text{bit tags}} \oplus \overbrace{\mathrm{AES\text{-}128_{k2}}(n)}^{\text{blinding tag}}$$

The second part of the formula XORs the blinding tag with the bit tags that correspond to the message. Here $|m|$ denotes the length of the message and $m[i]$ denotes the value of its i-th bit. Thus, all computations besides these final XOR operations can be done independently of the concrete message that will be authenticated. Therefore, BP-MAC's construction enables the quick computation of new tags, especially for short messages which require a low number of XOR operations.

As usual for deterministic MAC algorithms, the tag verification happens by computing the tag t^* for the received message m and then comparing it to the received tag t. If both tags are identical, the receiver can conclude, with high confidence, that the received message m has not been altered in transit.

4.2.4 Memory Optimizations for BP-MAC

The naïve realization of BP-MAC requires the storage of $2 \cdot |m|$ bit tags. For a fixed-length messages, we can, however, half this memory overhead while computing the same tag by precomputing the default tag of a message composed of only zeros. To support the same optimization for variable-length messages, we must pad the message before authenticating it. Then, we only need to store how to alter the tag for a bit set to one with so-called *bitflip* tags, of which we need $|m|$ in total (one

Algorithm 3: Memory-efficient signature generation Sig_k

Input: message msg, nonce $nonce$
Output: tag t

```
t ← AES_k2(nonce)                          ▷ Preprocessable blinding tag
len ← length(msg)                          ▷ The length of the message in bits
t ← t ⊕ t_default                           ▷ Compute tag of all-zero message
for n between 0 and len-1 do
    if the n-th bit in msg is set then
        t ← t ⊕ t_bitflips[n]               ▷ Lookup the cached change to t
    end
end
t ← t ⊕ t_bitflips[len]                     ▷ Padding for the message
return t
```

for each bit). We sketch the tag computation with this optimization in Algorithm 3, highlighting in blue the computations executed during the latency-critical phase. Already before a new message is ready, we can compute the blinding tag and XOR it with the default tag $t_{default}$ to generate the tag of an all-zero message as:

$$t_{default} = \bigoplus_{0 \le i < n} \text{AES-128}_{k1}(i||0)$$

Then, to compute the tag for the actual message, we need to change the tag for each bit that is not zero, as assumed for the default tag $t_{default}$. To efficiently realize these changes, we can precompute bitflip tags for all i bits in a message as:

$$t_{bitflip}[i] = \text{AES-128}_{k1}(i||0) \oplus \text{AES-128}_{k1}(i||1)$$

By XORing these bitflip tags to the default tag $t_{default}$, we compute the tag as if the bit at the corresponding position is set. Finally, we pad the message to a fixed length to be able to differentiate between a shorter message and one with trailing zeros. BP-MAC uses the *Padding method 2* as described by ISO/IEC 9797-1 [7], appending a 1-bit at the end of the message followed by as many 0-bits as necessary to reach the desired message length. As the default tag $t_{default}$ already assumes that all bits are zero, we only need to incorporate a single further bitflip tag t for the first index after the end of the message. Thus, we only need to store $|m| + 3$ tags (one for each bit, including padding, t_{nonce}, and $t_{default}$). This procedure reduces the number of operations during the latency-critical phase for each 0-bit, which accounts for about half of all bits if traffic is encrypted.

4.2.5 Online Computation of BP-MAC Tags

The design of BP-MAC enables another unique feature: online computability. Sometimes, the entire message is not available at once but instead is made available over time, *e.g.*, messages composed of multiple subsequently polled sensors or received

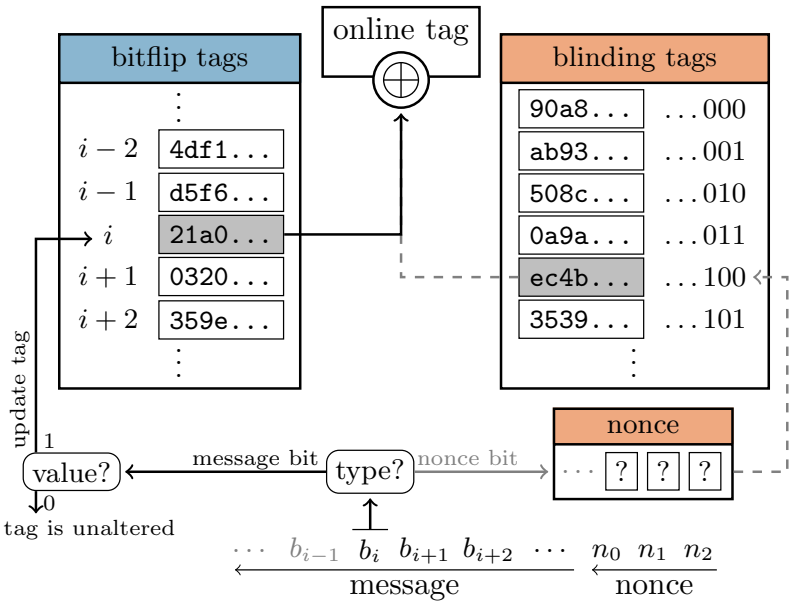


Figure 4.8 BP-MAC tags can be computed incrementally with each received bit, such that a single XOR operation suffices to compute the final tag once the final bit is read.

messages decoded bit by bit. In such cases, the computation and verification of BP-MAC tags does not have to wait until the entire message is available. Instead, tags can be computed bit by bit as those are made available. This adapted process to compute BP-MAC tags is illustrated in Figure 4.8.

For the online computation of a BP-MAC tag, the final tag variable is initialized with the default tag for a message composed of only zeros. The message is then received bit by bit. At the bottom of the figure, the bit b_i is currently processed. Its value is checked, and if the bit is set, the corresponding bitflip tag is XORed with the final tag. These bitflip tags are precomputed and stored alongside the key for the corresponding transmission source. Nonce bits are either explicitly transmitted or implicitly tracked by the sender and receiver. If transmitted, the (least-significant) nonce bits can be received and processed before or after the message bits. In Figure 4.8, we show an example where the last three nonce bits n_0, n_1, and n_2 are updated with the transmission, while the rest of the expected nonce is tracked implicitly. This update process allows self-synchronization in case of a short burst of failed transmissions. Once the complete nonce is known, the corresponding blinding tag is selected and XORed with the final tag. If the message and nonce bits are processed, the order of which actually does not matter, the final tag is known. Either way, the process after the last bit is received is composed of a simple conditional check and at most one XOR operation. Thus, BP-MAC enables the online computation of tags, which can save valuable time in latency-critical situations.

4.2.6 Security Discussion

To guarantee secure message authentication, BP-MAC utilizes the estbalished Carter-Wegman MAC construction (*cf.* Section 2.4.2) in which a tag t for a message m

and a nonce n is computed as $t = H_{k1}(m) \oplus F_{k2}(n)$, where F is a pseudorandom function covering the same output space as the universal hash function H. For F, BP-MAC uses the same AES-based procedure as UMAC [127], with the sole difference that it computes the result in advance. Meanwhile, the fragmentation of messages into individually authenticatable bits is a special case of the established universal hash function $F^{\oplus}$ [39] and guarantees *difference unpredictability* as the distance between two digests is always the combination of one or multiple unpredictable AES ciphertexts. Thus, BP-MAC is secure as long as the underlying cryptographic primitive (*i.e.*, AES) is secure. Moreover, we can easily exchange this primitive for BP-MAC if it is ever considered not secure enough. In the following, we define the considered threat model and subsequently discuss possible attack scenarios.

4.2.6.1 Threat Model

The attacker's goal is to alter traffic such that the recipient of a message accepts the modified message as genuine. To achieve this goal, the attacker can observe and alter the (plaintext) messages and exchanged tags to either learn the secret key used by BP-MAC or directly alter a message and the corresponding tag. The information to realize such an attack can be extracted directly from the observed traffic or through side-channel information by observing the timing of individual packets. However, we explicitly do not consider an attacker with (partial) control over one or both communicating entities that could try to access keying information through *e.g.*, cache side-channel attacks. However, BP-MAC does not prevent the use of common mitigation techniques that protect against such attacks [157].

4.2.6.2 Resilience to Key Recovery Attacks

BP-MAC is not susceptible to key recovery attacks, as otherwise a key recovery attack against the underlying cryptographic primitive, *i.e.*, AES, would exist. By overhearing transmissions, an attacker learns $t = H_{k1}(m) \oplus F_{k2}(n)$ for a known message m and nonce n. An attacker cannot learn $k2$ from such tags, as otherwise there would exist a key recovery attack against F_{k2}, *i.e.*, `AES-128`: After learning a digest $d = F_{k2}(m)$, the attacker could generate a key k', consider all unique messages m as nonces, and compute a BP-MAC tag t' for arbitrary new messages m' as $t' = H_{k'}(m') \oplus F_{k2}(m)$. Thus, if an attack that recovers $k2$ from t exists, this attack could be used to recover $k2$ from simple F_{k2} digests. Similarly, $k1$ cannot be recovered by an attacker. Even if an attacker would learn the output of $H_{k1}(m)$, a recovery of $k1$ would imply the existence of a key recovery attack against `AES-128`: The attacker can compute $H_{k1}(m_1||\cdots||m_k)$ for observed ciphertexts $ct = AES_k(pt)$ by considering the first part of an observed plaintext pt to be the index and the second part to be a truncated message, *i.e.*, $pt = i||m_i$. Then, if a key recovery attack would exist against H, this same attack would also enable recovering k for `AES` plaintext-ciphertext pairs. Hence, BP-MAC is at least as resilient against key recovery attacks as AES.

4.2.6.3 Unforgeability of Authentication Tags

However, a successful attack does not need to recover the authentication keys to attack a MAC scheme. In many cases, it suffices if an attacker can forge authentication tags and thus inject harmful messages into a protected communication. Nevertheless, BP-MAC is also secure against such attacks. First, a message is hashed by $F^{\oplus}$, with an unpredictable result for the attacker if they did not observe previous digests of $F^{\oplus}$. Therefore, each output of $F^{\oplus}$ is blinded by a unique pseudorandom number, ensuring that an attacker cannot learn a single output of $F^{\oplus}$. As the nonce changes for each message, no information about the internal state of BP-MAC is revealed to an outsider. Thus, BP-MAC is also secure in the presence of an attacker that merely wants to generate valid tags for altered or new messages.

4.2.6.4 Timing Side-Channel Attack against BP-MAC

The presented memory optimizations for BP-MAC execute some computations only for bits set to one, potentially enabling timing side-channel attacks. In particular, the time a message authentication computation takes might hint at the number of bits set to one in the authenticated message. If the message is sent in plaintext, this information is already accessible to third parties and does not leak confidential information or help in recovering keys. However, when authenticating encrypted traffic, it is crucial to operate in the *encrypt-then-MAC* mode [39] not to reveal information about the plaintext.

4.2.7 Performance Evaluation

BP-MAC reduces the delay of message authentication for short, mission-critical wireless communication. We validate this claim by performing measurements on two different embedded devices and evaluating BP-MAC's memory consumption.

4.2.7.1 Latency Measurements

We implement BP-MAC for Contiki-NG and evaluate its performance on two different architectures to ensure the observed benefits generalize across them: Zolertia Z1 (MSP430 @ 16 MHz, 16-bit CPU, 8 kB RAM) and Zolertia RE-Mote (ARM Cortex-M3 @ 32 MHz, 32-bit CPU, 16 kB RAM). To assess the performance of BP-MAC, we have to compare it to other MAC schemes with similar security guarantees, such as UMAC, VMAC, or Poly1305. We choose to compare BP-MAC against an optimized implementation of UMAC[3] as it introduces the least processing overhead for short messages (<32 bytes) even on an unfavorable architecture [126].

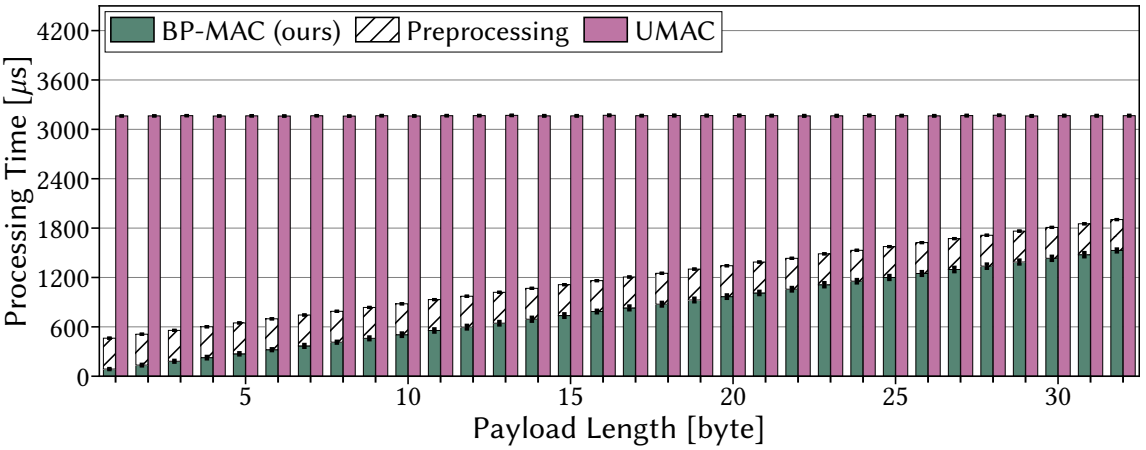


Figure 4.9 On the 16-bit architecture of the Zolertia Z1, BP-MAC outperforms UMAC by up to two orders of magnitude for short messages during the latency-critical phase and shows significantly less overall processing overhead for 32 byte long messages.

4.2.7.1.1 BP-MAC on the Zolertia Z1

First, we compare BP-MAC's and UMAC's performance on the 16-bit MSP430 processor of the Zolertia Z1, a typical processor for resource-constrained IIoT devices. Here, we report on the time for computing 16-byte tags for messages of varying lengths. Figure 4.9 depicts the means and 99% confidence intervals for the respective computations of one tag. We measure 100 tag computations and derive from this the time of one tag computation due to a too low clock resolution. We repeated each measurement 30 times. Note that Sig_k and $Vrfy_k$ require one tag computation each, such that the actual delay introduced into one secure transmission is twice the reported time.

As expected, the time required to compute UMAC tags is independent of the message length on the analyzed scale. In contrast, BP-MAC's bitwise processing introduces a linear dependency between message length and processing time. We observe that BP-MAC significantly outperforms UMAC, such that even for 32 bytes long messages, the overall processing time of BP-MAC is still 40 % lower. For 1 byte long messages, the difference is even more extreme, as BP-MAC induces a delay of only 86 μs for one tag computation, compared to 3.2 ms for UMAC. BP-MAC is thus significantly faster than UMAC for short messages on a 16-bit architecture.

We repeated the same measurements for shorter tags (12, 8, and 4 bytes). There, we observed similar behavior, whereas the absolute processing time of BP-MAC became shorter by $24.8 - 35.4\%$, $38.6 - 54.0\%$, and $51.1 - 66.5\%$ for increasingly shorter tags as shorter tags require fewer XOR operations. In turn, UMAC also becomes faster, primarily through its ability to reuse AES computations to mask multiple tags for 8 and 4-byte tags. However, the irregular need to compute AES blocks introduces jitter to the processing of UMAC, which is undesirable in low-latency communication [89, 233]. For BP-MAC, these bursty computations only occur in the preprocessing phase and thus do not influence actual transmission delays.

[3] `https://fastcrypto.com/umac`

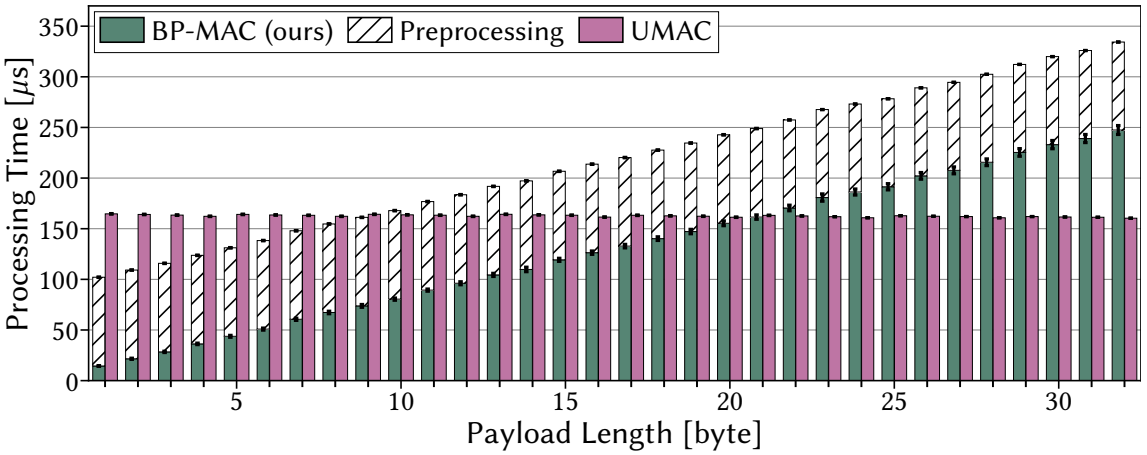


Figure 4.10 Even on the faster Zolertia RE-Mote, BP-MAC realizes faster tag computations than UMAC in the latency-critical phase for messages shorter than 21 bytes and less overall processing for messages shorter than 10 bytes.

4.2.7.1.2 BP-MAC on the Zolertia RE-Mote

To show the performance of BP-MAC on a more powerful device, we furthermore compare BP-MAC and UMAC on the Zolertia RE-Mote. While it still constitutes an embedded device, its ARM Cortex-M3 is significantly more powerful than the MSP430 processor of the Zolertia Z1. Also, its 32-bit processor is precisely the architecture targeted by UMAC [127]. Hence, we repeat our previous measurements and report on the results in Figure 4.10.

Still, BP-MAC outperforms UMAC for small messages, partially by more than one order of magnitude. To be precise, the latency-critical processing of BP-MAC is faster for messages not longer than 21 bytes. Furthermore, even the overall processing overhead is smaller for messages shorter than 10 bytes. These results thus show that BP-MAC is particularly suited for authenticating short messages, even in a worst-case comparison to UMAC. For shorter tags, this trend continues, with BP-MAC outperforming UMAC by $17.2 - 24.4\%$, $22.8 - 32.0\%$, and $35.2 - 48.3\%$ during the latency-critical phase for tags of sizes 12, 8, and 4, respectively. The tipping points below which BP-MAC outperforms UMAC are 26, 18, and 15-byte messages for increasingly shorter tags.

Concluding, BP-MAC enables secure communication based on the established security of AES with low processing overhead for small messages across different processor architectures. However, the performance gap between BP-MAC and UMAC is much narrower on the Zolertia RE-Mote. This behavior is not due to a worse performance by BP-MAC, but instead due to UMAC being specifically optimized for the 32-bit processor in this scenario.

4.2.7.1.3 BP-MAC and Hardware-accelerated Cryptography

Due to the rising importance of low-latency security in IIoT scenarios, low-powered devices increasingly provide hardware accelerators for cryptographic operations.

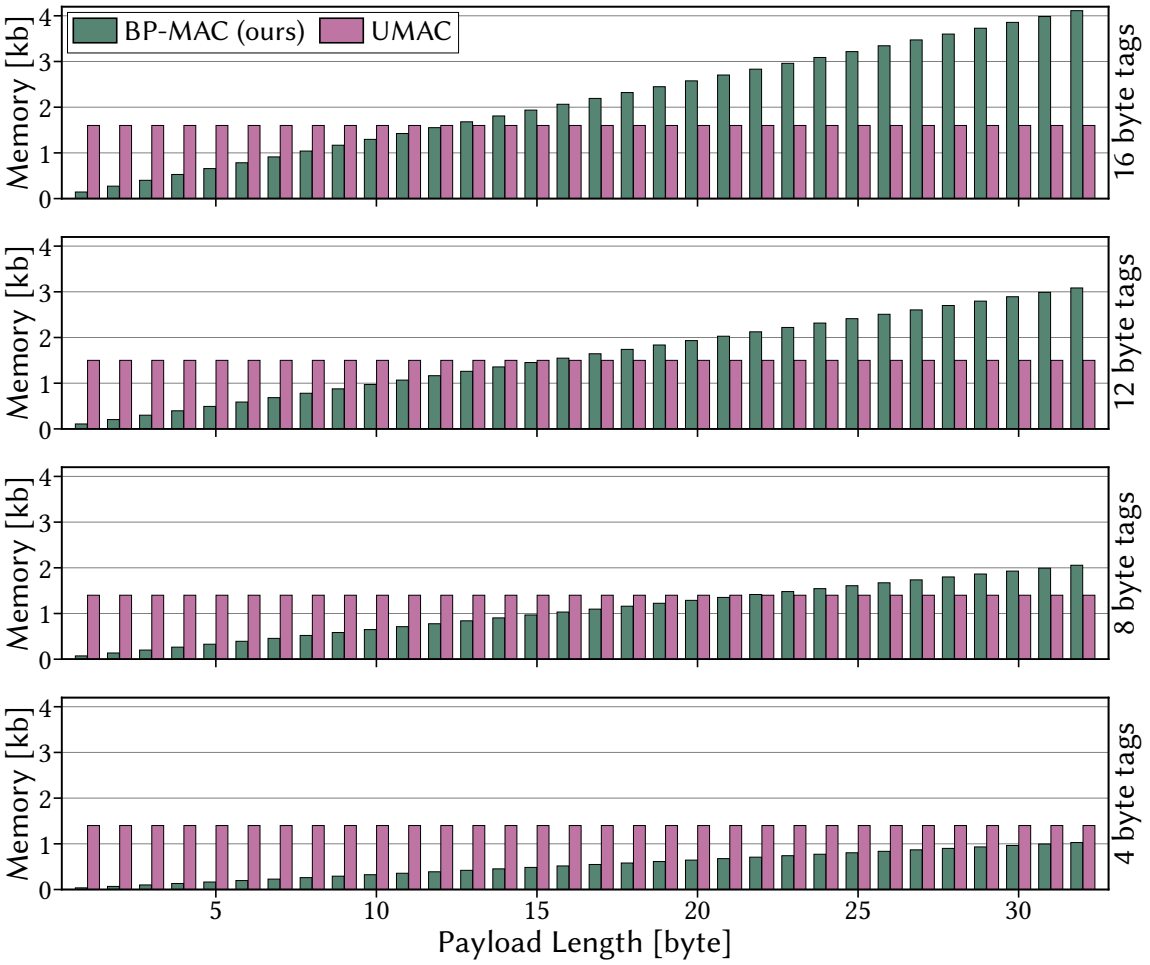


Figure 4.11 BP-MAC's memory footprint grows linearly with tag and message lengths. Thus, for messages shorter than 12, 15, 21, and 43 bytes, BP-MAC requires less memory than UMAC for 16, 12, 8, and 4-byte long tags, respectively.

Using the Zolertia RE-Mote's `SHA256` accelerator for HMAC computations shows that even hardware-accelerated cryptography introduces significant processing overheads. Indeed, our measurements reveal that the computation of one HMAC-SHA256 tag takes approximately 220 μs, independent of tag and payload lengths for small payloads. Thus, in these cases, hardware-accelerated HMAC-SHA256 is even slower than a software implementation of UMAC. Still, accelerators for specialized MAC schemes can be constructed to achieve even faster cryptographic operations. Fittingly, BP-MAC's heavy reliance on XOR operations and its high parallelizability by individually processing each bit promise highly efficient hardware acceleration.

4.2.7.2 Memory Overhead

Fast MAC schemes based on universal hashing, such as BP-MAC and UMAC, typically trade memory usage for high processing speeds, a limited resource on low-power devices. Hence, we compare BP-MAC's and UMAC's memory footprint by compiling Contiki-NG as a native Linux application. We then use Valgrind's massif tool to

analyze the peak memory footprint of both schemes for varying message and tag lengths. We depict our results in Figure 4.11.

We notice that the memory footprint of UMAC is constant for different message sizes and hardly changes across tag lengths (ranging from 1.4 kB to 1.6 kB). In contrast, BP-MAC's memory footprint increases with the sizes of tags and messages. Overall, BP-MAC's memory footprint increases by eight times the tag lengths when messages become one byte longer, which we expected since an additional bitflip tag has to be stored for each additional message bit. Consequently, BP-MAC's memory footprint is favorable for small messages up until a tipoff point where UMAC becomes, in turn, more resource-efficient. These tipoff points lie at a message size of 12, 15, 21, and 43 bytes for 16, 12, 8, 4 byte long tags, respectively.

Overall, our evaluation thus shows that BP-MAC outperforms UMAC in terms of processing latency and memory footprint for small messages while providing the same security guarantees, *i.e.*, reducible to the security of AES. Thus, BP-MAC enables secure communication in critical IIoT scenarios with a significantly smaller impact on communication latency than state-of-the-art approaches.

4.2.8 Summary

Some IIoT scenarios rely on low-latency communications with stringent security guarantees. In this context, a significant challenge are computationally intensive cryptographic computations of MAC schemes in the latency-critical phases, *e.g.*, once a new message is ready for transmission. While numerous MAC schemes aim to reduce latency, none focus on small messages of only a few bytes, an essential characteristic of many latency-critical IIoT scenarios. To fill this gap, we propose a new Carter-Wegman MAC scheme, BP-MAC, optimizing for the aforementioned small messages by the extensive use of preprocessing with a universal hash function. Thus, BP-MAC reduces latency-critical computations to a few XOR operations, which overall leads to more than a ten-fold reduction in processing overhead compared to the state-of-the-art. Consequently, BP-MAC enables integrity protection of small messages in latency-critical scenarios, even for Internet of Things (IoT) devices with limited computational and memory resources.

4.3 Conclusion

Processing overhead on resource-constrained hardware may lead to significant delays that can prevent the deployment of security mechanisms in IIoT scenarios. To address these concerns, we tackle potential bottlenecks on two layers. First, we design an efficient process to embed authentication data in a variety of legacy protocols with RePeL. Secondly, we design BP-MAC, a secure MAC scheme specifically speeding up the processing of short payloads as often encountered in IIoT scenarios.

RePeL demonstrates that even legacy protocols can be retrofitted with authentication data by repurposing unused header fields. The necessary processing may be

manageable by the IIoT devices themselves, depending on scenario requirements and available hardware. However, the embedding and verification of authentication data can also be offloaded to dedicated BitW modules, which, at the cost of slightly increased jitter, can integrate security at line rate for IIoT networks. Such a deployment can also be utilized to incrementally deploy secure communication without interfering with existing networks. Still, most latency is introduced by the processing of MAC schemes.

To optimize this overhead for the short messages often encountered in IIoT scenarios, we design BP-MAC. BP-MAC takes advantage of offloading processing into idling phases to reduce latency-critical delays when new data must be transmitted. Its design is optimized for short payloads of only a few bytes where it can outperform competing MAC schemes by over an order of magnitude. We thus demonstrate how the use of spare resources and optimizations for the specific types of messages encountered in the IIoT can significantly boost the performance of message authentication.

While we cannot eliminate the overhead of cryptographic processing in resource-constrained environments, our work shows that these limitations are often less restrictive than feared. In many cases, the selection of the security schemes with the right trade-offs for a specific scenario enables the deployment of strong security even in severly constrained environments.

5

> *I don't know half of you half as well as I should like, and I like less than half of you half as well as you deserve.*

Bilbo Baggins, *The Fellowship of the Ring*, 1954

Message Authentication for Group Communication Systems

Message Authentication Code (MAC) schemes are designed to protect the communication between two parties. As symmetric cryptography shares one key among all participants, the inclusion of a third party already prevents source authentication. In such a group communication system, the first party could generate a valid message masquerading as the second party and send it to the third party. This third party cannot determine if the message indeed originates from the second party or is spoofed by another owner of the (group) key. Typically, digital signatures would provide message authentication in such scenarios, although at the expense of significantly higher processing and bandwidth demands.

To unlock the efficiency of symmetric cryptography for group communication systems, we focus on specific use cases. While these use cases impose specific conditions that we can take advantage of to provide security, they are chosen to address acute real-world problems. First, we introduce the multicast source authentication scheme Caiba designed for bus networks. Caiba takes advantage of the broadcasting nature of buses, the fast and online computability of BP-MAC, and reactive overwriting capabilities to offer message authentication for group communication without the bandwidth overhead or delays induced by competing schemes. We show how Caiba is compatible with the AUTOSAR specification for Secure Onboard Communication (SecOC) [22] for the Controller Area Network (CAN) bus and thus offers the first multicast source authentication scheme deployable in cars and other vehicles.

Next, we investigate a second use case of group communication systems where unicast communication is extended by on-path middleboxes for specific tasks, *e.g.*, intrusion detection. For such scenarios, we design *Middlebox-Aware DTLS* (Madtls), an extension to the DTLS 1.2 protocol that enables the integration of middleboxes into an end-to-end secured channel. The protocol enables fine-grained read and write

access control for middleboxes down to granting access to individual bits of DTLS records. MADTLS can thus protect brittle Industrial Control System (ICS) and critical infrastructure networks against an ever-increasing number of cyberattacks. Overall, these two newly designed schemes demonstrate how the performance achieved by MAC schemes can be ported to group communication scenarios.

5.1 CAIBA: Multicast Authentication for Fieldbuses

The performance of BP-MAC and especially its online computability provide the foundation to design a new multicast source authentication scheme. The need for such a multicast source authentication scheme becomes apparent when looking at the lack of security in CAN bus networks. Virtually all motor vehicles currently on the roads are equipped with hundreds of small embedded computers, so-called Electronic Control Units (ECUs), that monitor and control vital vehicle functions [117]. To realize overarching functionality, these ECUs require means to communicate reliably with each other in harsh environments. The de facto standard for such in-vehicle interconnection is the CAN bus. Developed in the 1980s under the assumption of vehicles being physically isolated systems, security was not a design goal of CAN.

However, contrary to this original assumption, modern cars offer more and more (wireless) connectivity and are thus increasingly exposed to cyberthreats with potentially fatal consequences [55]. For example, a remote compromise of the infotainment system of a Tesla Model S enabled attackers to control the car's acceleration, leaving potential passengers at the attacker's mercy [178]. This example is not an isolated incident, but only one of many recent attack demonstrations [3,55,64,80,163,178,179,256]. Those attacks exploit that, once an ECU has been compromised, CAN provides no protection against ECU masquerading and arbitrary message spoofing [123]. Concerningly, the dark web offers car theft devices, the so-called CAN Invader, today that exploit the vulnerability of CAN [3]. These devices are actively used by criminals to steal modern cars [3].

The vulnerability of CAN has thus been exposed, while current efforts to retrofit security via intrusion detection or MAC schemes are insufficient as they cannot adequately protect against masquerading attacks, where a compromised communication device imitates another device. To remedy this situation, multicast source authentication is required to reliably identify the senders of messages. Here, the main challenge is the need for both reliable and immediate protection of multicast communication. Most notably, CAN is a broadcast communication protocol, where messages are often intended for multiple receivers [90]. Typically, to verify the source of a message in a multicast scenario, digital signatures are used as authentication and verification relies on different keys. However, digital signatures are not applicable in CAN due to excessive computational and bandwidth requirements [15]. CAN frames hold at most 8 bytes of payload, and even the up to 64 bytes of the CAN-FD extension are insufficient to support asymmetric cryptography.

To address the general lack of multicast source authentication schemes with adequate security properties, we hence design the Compact And Instantaneous Bus Authen-

tication (CAIBA) scheme. CAIBA relies on an authenticator reactively overwriting authentication tags on-the-fly, such that a receiver only reads a valid tag if not only the integrity of a message but also its source can be verified. We achieve interoperability with legacy CAN devices, while protecting receivers implementing the AUTOSAR SecOC standard against masquerading attacks without communication overhead or verification delays.

5.1.1 Background: CAN's Physical Layer

The CAN bus protocol is the standard in the automotive industry and mandatory in the EU for vehicle diagnostics and nowadays also finds applications in, *e.g.*, industrial automation [141]. To realize CAIBA, we need to dive into some of the less well-known details of CAN's physical layer, which we outline in the following.

5.1.1.1 Signaling

Physically, the CAN bus is based on two wires, CAN_H (CAN high) and CAN_L (CAN low), to which every ECU is connected [110]. At the physical layer, bits are encoded using a simple non-return-to-zero (NRZ) scheme via different voltage levels specifying the following differential coding [112]:

$$\begin{cases} 0 \text{ (dominant bit)} & \text{if } CAN\_H - CAN\_L \geq 0.9\,V \\ 1 \text{ (recessive bit)} & \text{if } CAN\_H - CAN\_L \leq 0.5\,V \end{cases}$$

We exemplify the resulting differential signaling in Figure 5.1. The transmitter only actively applies the voltage difference for dominant bits, while the recessive voltage levels resulting in a difference are passively created by internal resistance within each transceiver. Due to this wired AND gate (bus lines are recessive only when all transmitters transmit a recessive bit), dominant bits (0) always overwrite recessive bits (1) [110]. This property is utilized on the data link layer for arbitration, acknowledgments, and error handling.

5.1.1.2 Identifier and Data Frame Format

Two *data frame* formats with different Identifier (ID) schemes for addressing are standardized [110]. The base frame consists of an 11-bit ID with up to 8 bytes of payload, three control flags (3 bits), the length field (4 bits), the CRC (16 bits), and the Acknowledgement (ACK) field (2 bits) as shown in Figure 5.1. The extended frame format offers a 29-bit ID. Both data frames are enclosed in a start-of-frame (SOF) and an end-of-frame (EOF) delimiter, which are signaled by one dominant and 7 recessive bits, respectively.

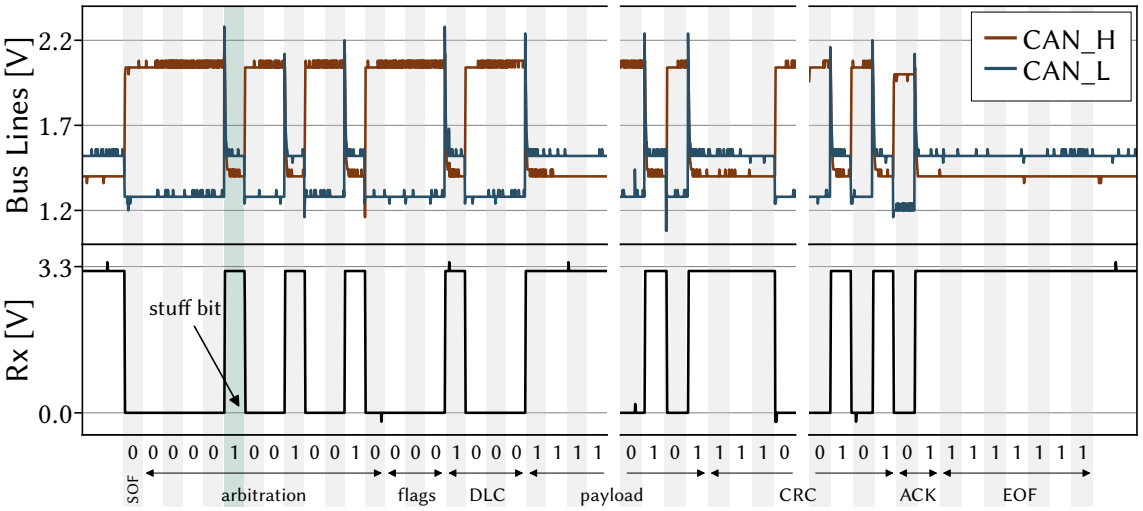


Figure 5.1 To transmit data, CAN uses differential encoding on two wires, CAN_H and CAN_L. We display a base format CAN frame with 8 bytes of payload as used to control hundreds of millions of cars every day. Note the gap in the payload and CRC field for increased readability.

5.1.1.3 Sampling and Synchronization

The fixed duration for which a single bit is present on the wire is called the *nominal bit time* and depends on the used data rate. Within this bit time, each signal is split into four time segments that consist of one or more *time quanta* which are derived from the ECU's clock and usually are configurable by a programmable prescaler [110]. The first segment is exactly one time quanta long and is used to synchronize the different ECUs. The second segment compensates for signal propagation and its length, in combination with the bit rate, are thus the main contributors to determining the maximal bus length. The last two segments are chosen such that the bit sampling, happening between them, is located as close as possible to 75 % of the nominal bit time [78]. These two segments can also be elongated or shortened for resynchronization based on continuous monitoring of the edges of the voltage levels.

5.1.1.4 Bit Stuffing

With NRZ coding, a certain number of consecutive bits of equal value leads to an absence of edges necessary for resynchronization, which is addressed by *bit stuffing*. To prevent more than five identical bits appearing in sequence in a bit stream, an additional bit with the inverse level is inserted (*stuffed*) after five identical bits (cf. Figure 5.1). Stuff bits are therefore transparently added and removed during transmission by the transmitting and receiving CAN controllers.

5.1.2 State-of-the-Art on Securing CAN

Historically, the CAN bus protocol offers no protection against cyberattacks. Recent real-world demonstrations have shown that this lack of security enables the remote

compromise of entire vehicles [3, 55, 64, 80, 163, 178, 179, 256]. A famous example takes full control of Tesla Model S by remotely attacking the car's multimedia system and propagating from there through the CAN bus [178]. Consequently, new protection mechanisms are required as preparation for the ever-increasing digitalization and interconnection of modern cars.

5.1.2.1 Threat Model

We consider a threat model that corresponds to that chosen in the majority of the recent offensive and defensive research on CAN security [28, 36, 55, 80, 152, 163, 178, 179, 183, 190, 261]: We assume that an attacker has either physical access to the bus to implant a malicious ECU or has compromised an existing ECU, *e.g.*, through Bluetooth or other connectivity. The attacker then intends to inject messages beyond those message types needed for the functionality of the compromised ECU (an implanted malicious ECU is not supposed to send any messages), *i.e.*, masquerading as another ECU, to highjack the car.

5.1.2.2 Related Work

Research on protecting CAN against malicious ECUs can be grouped into three categories: *intrusion detection systems (IDSs)*, *covert channels*, and *cryptographic approaches*. These approaches make certain trade-offs to achieve alleged security, which result in certain limitations, vulnerabilities, or overhead. We classify the current state-of-the-art in protecting CAN according to these drawbacks as introduced in the following.

Limitations

- Only point-to-point communication is protected, whereas broadcast communication, *i.e.*, CAN messages meant for multiple receivers, are not protected.
- Only broadcast messages with few (*e.g.*, < 5) receivers are protected, where the level of protection exponentially reduces with the number of receivers.
- Definitively lost CAN frames lead to desynchronizations that cannot be recovered from.
- The protocol deterministically delays message authenticity verification by buffering messages at either the sender or receiver.

Vulnerabilities

- The physical disconnection of a single ECU (*e.g.*, an IDS) covertly disables all protection mechanisms in a way that is unnoticeable by any other ECU.
- An attacker that compromises a single CAN ECU can subsequently masquerade as other ECUs, *e.g.*, because all ECUs share a group key. Needing to compromise a dedicated security node (*e.g.*, IDSs) is considered more secure, as those offer no outside connectivity and should be temper-resilient.

- An attacker can intercept a frame and use the information gained to impersonate another device, *e.g.*, by replaying the frame at a later time. Some approaches may only enable the injection of malicious frames shortly after the original intercepted frame.
- The detection of malicious messages only happens retroactively after messages have been processed and potentially caused significant harm.

Overhead

- The scheme needs excessive storage overhead, *e.g.*, to store a secret key for each other ECU.
- The scheme requires some additional communication overhead, either by reserving some space in each frame or even transmitting additional CAN frames.
- Frame transmissions are delayed, or prioritization is not fully respected.

We consolidate our analysis of the state-of-the-art on protecting CAN in Table 5.1. Our classification is partially based on a recent survey by Lotto *et al.* [152] on weaknesses in CAN authentication protocols. For the five proposals classified as *secure* by Lotto *et al.* and seven proposals not considered in their survey, we present a detailed analysis in Appendix A to substantiate our classification. In the following, we give an overview of the three categories of CAN protection mechanisms.

Intrusion Detection Systems (IDSs) add a dedicated device to the network that monitors the physical characteristics and behavior of all ECUs to detect impostors. This monitoring can be based on voltage levels [58,59,79,119], message timings [57,177, 206,214,224,228,252,265], signal characteristics [118,150,177], or device behavior [149, 170]. All methods have in common that the IDS device can be disrupted to covertly disable security without this being noticed. Moreover, IDSs often only detect malicious message flows rather than achieving single-message detection, such that timely reactions, *e.g.*, discarding messages, are hardly possible [219]. Finally, IDSs have no perfect detection performance, *i.e.*, malicious messages may not get recognized (false negative) or genuine traffic falsely gets flagged (false positive). Even low false positive rates can have detrimental effects if acted upon, given the amount of genuine CAN messages being constantly sent. On top, recent research shows how sophisticated attacks can also deliberately evade IDSs in CAN [36, 210]. Overall, we conclude that IDSs may be useful, but should not be the only line of defense due to their limitations.

A second class of CAN security approaches relies on *covert channels* to achieve message authentication. These covert channels can be created through a high-frequency signal interlaced with regular messages [91, 162, 184], which, however, requires dedicated transceiver hardware for decoding and still does not provide source authentication. ZBCAN [219], on the other hand, proposes unique inter-frame spacings for each sender, only known to the sender and a central authority. In case of anomalies, the central authority jams the suspicious frame. However, in ZBCAN, it is neither detectable that the central authority is covertly disconnected nor is the protocol secure against bit modification attacks [137] as it only verifies that the sender

	Scheme	Year	Limitation				Vulnerability				Overhead		
	IDSs	2016-2023	-	-	-	-	●	○	-	○	-	-	-
Covert Channels	CANTO [91]	2020	-	-	-	-	-	●	-	-	-	-	●
	Watermarking [162]	2022	-	-	-	-	-	●	-	-	-	-	-
	ZBCAN [219]	2023	-	-	-	○	●	-	○	-	-	-	○
	CAN-MM [184]	2024	-	-	-	-	-	●	-	-	-	-	-
Cryptographic Approaches	AUTOSAR SecOC [22]	2020	-	-	-	-	-	●	-	-	-	○	-
	CANAuth [241]	2011	-	-	-	-	-	●	-	-	●	○	-
	Car2X [217]	2011	-	-	-	●	-	-	●	-	○	●	-
	LiBrA-CAN [90]	2012	-	●	-	●	-	○	-	-	-	●	-
	LinAuth [147]	2012	-	●	-	-	-	-	-	-	●	●	-
	LCAP [100]	2012	-	-	-	-	-	-	●	-	●	●	-
	CaCAN [131]	2014	-	-	●	-	●	○	-	-	-	○	-
	Woo-Auth [261]	2014	-	-	●	○	-	●	-	-	○	○	-
	VeCure [253]	2014	-	-	●	○	-	●	-	-	●	●	-
	LeiA [197]	2016	-	-	-	●	-	●	-	-	●	●	-
	vatiCAN [183]	2016	-	-	○	●	-	●	○	-	○	○	-
	VulCAN [240]	2017	-	-	-	●	-	●	-	-	●	●	-
	TOUCAN [28]	2019	-	-	-	-	-	●	-	-	-	○	-
	LEAP [153]	2019	●	-	-	-	-	-	-	-	●	○	-
	CAN-TORO [92]	2020	-	-	-	-	-	●	○	-	●	-	-
	MAuth-CAN [113]	2020	-	-	○	●	●	○	-	-	-	○	-
	AuthentiCAN [159]	2020	●	-	-	-	-	-	-	-	●	●	-
	S2-CAN [190]	2021	-	-	-	-	-	●	-	-	-	○	-
	Caiba	2025	-	-	-	-	-	-	-	-	-	○	-

● full ○ partial
Limitation: no broadcasting support, limited scalability, no resynchronization, delayed verification
Vulnerability: disconnected device, masquerading by other ECU, frame interception, delayed alarms
Overhead: storage overhead, communication overhead, delayed transmission

Table 5.1 Current proposals to protect CAN traffic expose weaknesses that make them either not deployable in cars (*e.g.*, scalability limitations) or offer attack vectors to malicious actors (*e.g.*, no protection against masquerading).

intended to send at a given time but not the content of the message. Moreover, ZBCAN modifies CAN's prioritization scheme, which can lead to a significant delay of up to 25 ms.

Finally, *cryptographic approaches* rely on the integration of integrity-protecting tags, usually MACs, into data frames. These tags can be transmitted *e.g.*, as a portion of the payload [22], as a substitution for the Cyclic Redundancy Check (CRC) checksum [261], as part of the CAN ID [92], or in an additional frame [183]. A prominent example is the AUTOSAR SecOC standard [22] that uses part of the

payload, *e.g.*, 28 bit, for the transmission of integrity protection. Like AUTOSAR SecOC, most of these approaches rely on *group keys* to compute integrity tags. The resulting lack of source authentication means that compromised ECUs are not restricted from simply impersonating other devices. The few exceptions split the limited space for integrity protection among all receivers [90, 147], do not support broadcast communication [153, 159], rely on a covertly disconnectable authenticator [113, 131], or are vulnerable to frame interceptions [100, 217].

Concluding, we observe that current proposals to protect CAN are either vulnerable or unsuitable for in-vehicle deployments. With CAIBA, we aim to design a protocol without these drawbacks.

5.1.3 Source Authentication with CAIBA

For traditional point-to-point communication, source authentication and integrity protection are realized by appending an integrity-protecting authentication tag (*i.e.*, a MAC) to each message. This tag is computed with the help of a secret key shared by the sender and receiver, and verified by the receiver upon the reception of a message. However, in a broadcasting domain like CAN, such authentication tags do not provide source authentication: A receiver cannot differentiate whether a message stems from the alleged sender or if it has been spoofed by another member in the group of receivers who has access to the secret key. Hence, a compromised device could masquerade as one or multiple other devices. To protect the CAN bus against masquerading attacks, an effective source authentication protocol for multicast communication is thus required. In the following, we first investigate existing proposals to achieve source authentication and argue why they are not suitable for CAN. Afterward, we propose our novel scheme, CAIBA, that addresses limitations of prior work.

5.1.3.1 The Current State-of-the-Art

Source authentication for multicast communication relies on asymmetry between senders and receivers or across receivers [54]. Digital signatures, where receivers only have keys to verify messages, but not authenticate them, are the most widespread form of asymmetry. The main concern for this type of *cryptographic asymmetry* is the size of signatures (upwards of 32 bytes) that cannot fit into CAN frames. The TESLA protocol [188, 189], on the other hand, relies on *time asymmetry* for multicast source authentication. Here, keys are revealed after they have expired and linked to the sender (*e.g.*, via a hash-chain), such that receivers can verify the authenticity of buffered messages retroactively. Time asymmetry does, however, introduce significant delays and some additional bandwidth overhead (*e.g.*, for key revelation) which are not acceptable in in-vehicular communication. The final type of multicast source authentication schemes relies on *information asymmetry*, where receivers can only verify parts of the multiplexed source-verifying information [44, 187]. Thus, each receiver has reduced certainty of a message's authenticity, with the benefit that no single receiver can generate an authentication tag that is accepted by all receivers.

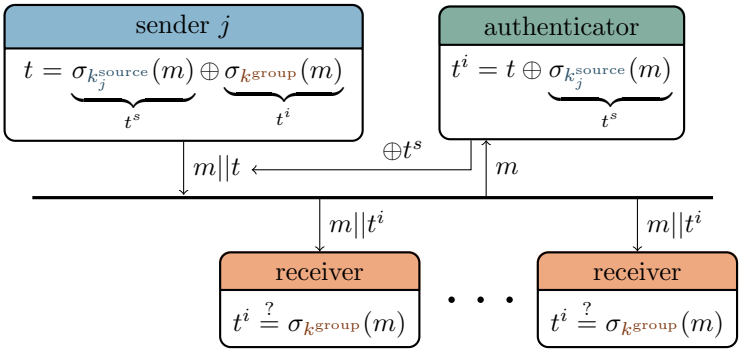


Figure 5.2 CAIBA XORs an integrity-protecting tag t^i and a source-authenticating tag t^s that are computed by the MAC function σ using the source and group keys, k_j^{source} and k^{group}, to protect a message m. During transmission, the authenticator computes t^s and overwrites the message such that only t^i remains. The integrity-protecting tag t^i that is read by the receivers can then be verified with a conventional group key without knowledge of the source key.

This security, however, comes at the cost of longer authentication data (compared to simple MAC schemes), which grows with the number of receivers. To conclude, current multicast source authentication schemes are inapplicable to CAN: they either incur verification delay or require excessive message overhead (in CAN, payload and authentication tag must fit into 8 bytes).

5.1.3.2 General Idea of CAIBA

To enable multicast source authentication for CAN, we devise the novel CAIBA scheme. To understand the idea behind CAIBA, consider the following hypothetical scenario. A CAN message is protected by two authentication tags, the *integrity-protecting tag* t^i and the *source-authenticating tag* t^s. These tags are computed with traditional MAC schemes, where the tag verification corresponds to the re-computation of the tag based on the received data and comparison to the received tag. The integrity-protecting tag t^i is generated using a group key k^{group} and verified by each receiver. Like intrusion detection systems and other proposals to retrofit integrity protection into CAN [90, 131, 219], CAIBA relies on one or multiple dedicated security nodes, the authenticator(s). The source-authenticating tag t^s is only verifiable by this authenticator that shares a symmetric key k_j^{source} with each sender j, but does not know the group key. In this hypothetical scenario, the authenticator also controls a virtual side-channel to securely notify each receiver whether or not the source of a message has been correctly verified.

Obviously, this basic construction occupies additional space (to send two tags instead of one) and time (to wait for the confirmation of the source-authenticating tag t^s verification to receivers). However, in CAIBA, we only transmit a single tag and use an implicit side-channel as shown in Figure 5.2: The actual transmitted tag t is the aggregation of both tags, *i.e.*, $t = t^i \oplus t^s$. The authenticator no longer verifies t^s but only recomputes t^s based on the received payload and the alleged sender. The authenticator then XORs the tag t with t^s as the bits are transmitted over the bus. When the ordinary receivers in CAN sample the bus, they will then read and verify

t^i. If, and only if, neither the data nor the tag has been manipulated beyond the authenticator's actions, the receiver's integrity verification can be successful. In this way, no explicit side-channel communication is needed, and only the space of a single authentication tag is occupied in the CAN payload.

5.1.3.3 Requirements to Deploy Caiba

Since CAIBA introduces no verification delay or message overhead compared to an ordinary group key-based authentication, our scheme is especially attractive for the CAN bus. However, as other solutions for multicast source authentication, CAIBA can only be applied if certain requirements are fulfilled. Specifically, the deployment scenario must fulfill certain requirements.

No Direct Communication. Entities must not be able to communicate directly with each other without the authenticator also overhearing these transmissions. Otherwise, a malicious group member could only append the integrity-protecting tag t^i, and a receiver would have no way of knowing whether the message really stems from the alleged transmitter. Buses like CAN fulfill this requirement by design.

Ability to Overwrite Messages. The authenticator must be able to reliably overwrite messages at line rate. We later show how this is possible for CAN. For other protocols, it still has to be investigated how to unlock such capabilities.

No Authenticator Collusion. There must be no collusion between the authenticator and any other entity in CAIBA. Otherwise, the entity could transmit a message without a source-authenticating tag, and the colluding authenticator would simply ignore it. Practical demonstrations have shown how to compromise individual CAN ECUs due to external connectivity. However, additionally compromising an authenticator with no external connectivity that can rely on temper-resilient hardware would require a significantly more advanced attack.

5.1.4 Security of CAIBA

In the following, we look at the security of CAIBA. Therefore, we first formally prove the security level achieved by the integrated authentication tags. Afterwards, we discuss the resulting security properties.

5.1.4.1 Security Proof

In this section, we want to formalize the security of CAIBA. CAIBA's security relies on two key assumptions towards an *adversary* $\mathcal{A}$:

- $\mathcal{A}$ cannot simultaneously compromise an ECU and the authenticator.
- $\mathcal{A}$ can only undetectably overwrite bits if they compromise the authenticator.

From these two assumptions, we can prove that CAIBA achieves the same security levels as an AUTOSAR SecOC instance with the same tag length. Here, we assume that the underlying MAC schemes are deterministic (m has exactly one valid t) and ideal (an adversary's best strategy is to guess a valid tag with a success rate of $1/2^{|t|}$, where $|t|$ is the bit-length of t). Thus, *e.g.*, 3-byte long tags lead to a 1 in 2^{24} chance of a tag being misclassified as valid, *i.e.*, a security level of 24 bit, for CAIBA and AUTOSAR SecOC alike.

We prove the security of CAIBA with a game as typically done for MAC schemes [39]: $\mathcal{A}$ may query an oracle with messages $m_i \in \mathcal{M}$ for t_i and eventually outputs a candidate forgery $(m', t'), m' \notin \mathcal{M}$, where $\mathcal{M}$ is the message space. $\mathcal{A}$ wins this game if the tag t' is valid for the message m'. The security of the scheme is then expressed as $\mathrm{P}[\mathcal{A}\text{ wins}]$, i.e., the probability that $\mathcal{A}$ wins this game.

We have to consider two cases. First, an adversary may have compromised an ECU. In this case, $\mathcal{A}$ can generate t^i but not t^s (unless the compromised ECU is authorized to send CAN ID). $\mathcal{A}$ may query the oracle for t_i^s or t_i for $m_i \in \mathcal{M}$. However, to interfere $t^{s\prime}$, $\mathcal{A}$ has no better strategy than guessing, as the underlying MAC scheme is considered secure. Meanwhile, there exists no better strategy than guessing t' either, as otherwise the MAC scheme to compute t^s would not be ideal (*cf.* Section 2.5). As the receiver expects to receive $t^{i\prime}$, *i.e.*, t must be $t^i \oplus t^s$ before modification, $\mathcal{A}$'s best strategy is to randomly guess a tag, i.e., $\mathrm{P}[\mathcal{A}\text{ wins}] = 1/2^{|t|}$.

Secondly, an adversary may have compromised the authenticator. In this case, $\mathcal{A}$ knows the keys to compute all source-authenticating tags t^s, but cannot compute integrate-protecting tags t^i. $\mathcal{A}$ could inject a frame without overwriting it. However, therefore they would need to guess a valid t^i, which only succeeds with a probability of $1/2^{|t^i|}$. Alternatively, $\mathcal{A}$ could modify a transmitted message to modify its origin (CAN ID) or content. However, if any of these two fields are modified, t^i is no longer valid and $\mathcal{A}$ would need to guess a new valid tag.

In both cases, $\mathrm{P}[\mathcal{A}\text{ wins}] = 1/2^{|t|}$. Thus, the security of CAIBA depends on $|t|$. In practice, MAC schemes are not ideal, so the MAC scheme chosen in a concrete deployment will cause marginally lower security.

5.1.4.2 Discussion of Security Properties

As proven in the previous section, CAIBA provides the same security as a traditional MAC tag of the given length used to protect an end-to-end connection between sender and receiver. Considering an exemplary 24 bit tag, an attacker thus has a mere $1/2^{24}$ (~1 in 17 million) chance to guess a valid tag. This security level is achieved because the aggregation of the source-authenticating tag and the integrity-protecting tag achieves the same security level as the individual tags [115]. The difference is that the aggregated tag can only be verified by combining the knowledge to verify both tags individually. As only the sender knows the keys for both tags, a valid tag authenticates the source of a message to each receiver.

One security risk for CAIBA is that an attacker physically disconnects the authenticator from the network. With other security solutions, *e.g.*, IDSs in CAN, such attacks

are often not detectable, and the attacker would gain complete control over the bus. In contrast, CAIBA quickly detects that the authenticator is disconnected and can take corrective actions before significant damage can be caused. Ideally, redundant authenticators are available, which can quickly take over upon request by the sender that noticed an inactive authenticator. Alternatively, operators can be notified about the inoperational authenticator and respond similarly, as to an alarm by an IDS. Luckily, an authenticator's physical disconnection most likely occurs while the car is stationary, such that a car could be prevented from moving, at least until the driver is notified about the risk. If the authenticator nonetheless disconnects while the car is moving, CAIBA could fall back to insecure CAN, advise the driver to stop the car, and potentially the speed and acceleration of the car could be limited.

Finally, an attacker could compromise the authenticator directly, *e.g.*, through a supply chain attack. However, even a compromised authenticator cannot spoof messages as it cannot compute integrity-protecting tags. To successfully spoof a message, an attacker needs to additionally compromise a genuine receiver within the corresponding multicast group. Thus, CAIBA overall provides strong protection against masquerading attacks and enables the detection of a disconnected authenticator.

5.1.5 Integrating CAIBA into the CAN Bus

After presenting the idea of CAIBA in a general and abstract fashion, we now discuss the technical details of integrating CAIBA into CAN. Here, we are faced with two main challenges. First, the authenticator must be able to quickly compute the source-authenticating tag t^s as the first bit of the overwritten tag follows immediately after the last bit of the protected payload. Secondly, the authenticator must be able to dynamically and precisely flip individual bits. In the following, we will successively introduce the design of all entities, *i.e.*, the transmitter, the authenticator, and the receiver. Before, we shortly address the general challenges of deploying CAN extensions such as CAIBA. Concerning key distribution, we assume that each ECU and the authenticator are initially configured with exactly those keys that they require, *i.e.*, each ECU shares a unique key with the authenticator, and all ECUs know relevant group keys[1].

5.1.5.1 Deployment Considerations

CAN extensions, such as CAN-FD for higher bandwidth, have in the past been successfully deployed by ensuring interoperability with legacy CAN devices. Thus, while ECUs do not necessarily need to be fully compliant with the existing CAN standard, any changes must be compatible with the existing standard, such that CAN ECUs and CAIBA ECUs can operate on the same network. Only then can an incremental deployment of CAIBA be possible, as the sudden adoption of a new standard by all actors in, *e.g.*, a car manufacturing supply chain, is not realistic. The easiest solution to achieve such interoperability is if all changes are restricted to the

[1] There maybe only exists a single group key for the entire bus.

inner operation of an individual ECU, while the signals written to the bus are fully compliant to the CAN standard at all times.

Regarding the actual integration and commercialization of CAIBA into *e.g.*, cars, ECU and car manufacturers do not have to change much in their operations. Nowadays, CAN controllers are mostly integrated as Semiconductor Intellectual Property (SIP) cores (e.g., [1, 4]) Once a SIP core implements CAIBA, ECU manufacturers can integrate them into newly manufactured chips. Then, car manufacturers need to add an authenticator to their bus and configure it for CAIBA-supporting ECUs to talk securely if all receivers interpreting a specific frame implement the AUTOSAR SecOC standard.

5.1.5.2 Transmitter Design

Transmitters of CAN frames only require minor changes to support CAIBA. These changes, however, require little additional space on the die of the ECU chip.

The main change between a transmitter supporting the AUTOSAR SecOC standard and a transmitter supporting CAIBA is the computation of two authentication tags instead of one over the concatenated CAN ID and payload. For the source-authenticating tag t^s, we rely on BP-MAC as it is optimized for short messages of only a few bytes. The integrity-protecting tag t^i can be computed by any suitable MAC scheme, which also allows the deployment of CAIBA without any modifications to the receivers, as they only receive and verify this tag. Once both tags are computed, the transmitter simply aggregates them with XOR for the transmitted tag t and integrates it into the payload of the CAN frame. We use 24 bits reserved in the payload for the tag and keep track of a counter for replay protection by appending its 4 least-significant bits in each frame. The usage of a 24 bits MAC with the 4 least-significant counter bits corresponds to `SecOC Profile 3 (JASPAR)` [22].

Furthermore, the transmitter must be adapted to support tag overwriting by the authenticator. On the one hand, the CRC checksum and the placement of stuff bits in the final received frame must be ensured. Therefore, the sender computes the checksum and the stuff bits placement based on the expected final frame, after the authenticator's modification, and not based on the transmitted frame. The authenticator knows when to expect stuff bits and skips over them. For the particular CAN controller used for our implementation, this requirement was naturally fulfilled: The controller listens to the bus while transmitting its message to detect higher-priority transmissions and uses this received data for bit stuffing and CRC computation.

On the other hand, the transceiver must ignore overwritten signals during the transmission phase of the tag. Otherwise, the transmitter would switch to the bus-off state and virtually disconnect itself from the bus. Therefore, we alter bit monitoring to listen to the expected bits after an authenticator overwrote the bits. This change does not lead to security or reliability problems, since every unauthorized modification would result in an invalid tag on the receivers' side. If errors only increase during the tag transmission, then an ECU can conclude that the authenticator is faulty or disabled and react accordingly. This reaction could be the fallback to unsecured CAN after alerting the network or the switch to a backup authenticator.

While CAIBA requires some modifications to CAN transmitters, it is important to note that the communication on the bus still remains CAN compliant and interoperable. Therefore, CAIBA can coexist with legacy CAN transmitters as long as the authenticator knows which communications are CAIBA-protected and which are not.

5.1.5.3 Authenticator Design

The authenticator continuously monitors the bus and notices when the transmitter starts writing to the bus. Then, it has to *(i)* identify the transmitter, *(ii)* compute the source-authenticating tag t^s over the CAN ID and the first part of the payload as well as *(iii)* overwrite the second part of the payload based on that computed tag. All of this processing needs to happen concurrently with the ongoing transmission of the CAN frame. In the following, we discuss how CAIBA realizes these three steps.

5.1.5.3.1 Source Key Identification

The authenticator first identifies the alleged source of a frame and whether it supports CAIBA[2]. While CAN uses message identifiers that should be associated with a unique sender, often employed standards such as the J1939 protocol ensure an easy coupling without a large lookup table [152]. Thus, the message ID can be used to identify the unique transmitter of a message. It is crucial to securely configure which ECU supports CAIBA during vehicle assembly to thwart downgrade attacks. After the sender has been decoded by the authenticator, the transmitter and thus the relevant key k_j^{source} is available.

5.1.5.3.2 Fast In-Line MAC Computation

When a CAIBA transmitter is identified, the authenticator computes the source-authenticating tag t^s based on the source authentication key k_j^{source}, the protected payload, and a nonce (a 8 bytes long counter which is synchronized based on the 4 least-significant bits transmitted with each frame). Considering CAN's maximum supported data rate of 1 Mbit/s, the first bit of the tag follows within 1 µs after the last data bit. Actually, the authenticator needs to compute t^s in a fraction of this time, as it needs time to overwrite this first tag bit before it is read by the receiver.

To achieve these speeds while still relying on sound cryptography, CAIBA uses the BP-MAC scheme (*cf.* Section 4.2) with its unique performance through bit-wise precomputing tags. The computation of the source-authenticating tag t^s can thus be achieved with a single XOR operation per read bit, which is fast enough to be performed even in a fraction of the time between two CAN signals. The authenticator is thus fast enough to know the source-authenticating tag t^s that it must XOR with the transmitted tag t as soon as its first bit is written to the bus.

[2] We assume that the bus is configured according to AUTOSAR SecOC, *i.e.*, space for authentication tags is reserved and nodes not capable of performing the integrity verification ignore the tag. Alternatively, all ECUs must track for which CAN IDs integrity protection is enabled.

5.1.5.3.3 Overwriting CAN Frames

Once the authenticator has computed the source-authenticating tag t^s, it must XOR this value with the transmitted tag, *i.e.*, whenever a bit in t^s is set, the signal on the bus must be inverted. For the signal-flipping procedure, the authenticator first needs to receive the transmitted signal. By sampling early in the nominal bit time, the authenticator can read the original signal while still having time to react. This early sampling is similar to the process by which CAN-FD achieves higher bitrates than CAN. It is possible because of relatively large tolerances for rise and propagation times in the CAN standard, which are designed for the worst case where the transmitter and receiver are located as far apart as possible. We suggest that the authenticator is placed centrally for bus length close to the maximum supported for a given bitrate, such that its distance from the transmitter is at most half of what is compensated for by CAN (even less if multiple authenticators are used). If the bit on the bus is recessive, the authenticator can simply overwrite it with a dominant bit. However, the opposite situation is not trivial, as dominant bits always overwrite recessive ones. For this purpose, we take advantage of the limited output current of CAN transceivers by simply connecting an additional *inverted* transceiver in our Reactive Bit Flipping (RBF) approach, which is explained in more detail in Section 5.1.6. With RBF, the authenticator can ultimately XOR the transmitted tag t with the computed tag t^s, such that receivers sense the integrity-protecting tag t^i when sampling the bus at standard sampling points.

5.1.5.4 No Need to Adapt Receivers

The design of CAIBA requires no software or hardware changes at the receiver if the AUTOSAR SecOC standard is already supported, since MACs are already implemented. The modifications of CAIBA only overlay the traditionally transmitted integrity-protecting tag t^i with the source-authenticating tag t^s. However, this change is transparent for an ECU which samples the bus as defined by the CANopen standard [78]. Thus, receivers do not need to be changed or replaced to support CAIBA, which enables a smooth and iterative deployment of CAIBA.

Once a car manufacturer integrates an authenticator module, all CAN transmitters can decide to employ CAIBA. Each sending ECU employing CAIBA needs to share a secret key k_j^{source} with the authenticator, which will, in most cases, be statically configured by the manufacturer. Afterward, the authenticator starts overwriting the authentication tag according to CAIBA's design. A receiver can then process CAN frames exactly as before. If a message is tampered with, either through manipulations during transmission or by being sent from an unauthentic source, the verification by the receiver fails, and it processes this anomaly accordingly.

5.1.5.5 Error Handling and Recovery

Failed CAN transmissions could result from desynchronized nonce counters of the sender, authenticator, and receiver. If this were the case, no future message would

be verifiable. However, if CAN transmissions are not received by all ECUs, *e.g.*, due to an error in the CRC checksum at a single ECU, this is announced with a distinct error frame. In that case, the authenticator resets its counter, and the sender retransmits the frame with the original MAC. The receiver would not have processed the MAC as the error frame interrupts the transmission, so no action is required.

Due to the transmission of the four least-significant bits of the nonce in each frame, it is unlikely that a node gets out of sync. To get to such a state, one ECU must miss an error frame or a frame must be discarded due to multiple failed retransmissions, at least 16 times in a row from the same origin. If such an unlikely scenario were to occur, it most likely stems from a malfunctioning ECU or an active denial of service attack, both of which CAIBA cannot protect against. Still, to ensure the best possible resilience, CAIBA implements a distinct recovery mechanism for such cases. If a receiving node fails to verify the tag of five consecutive frames, it requests a counter reset through a specific CAN ID.

Once this request is received by the sending ECU, it updates the two counters for the integrity-protecting and source-authenticating tags. Therefore, the value in the n most-significant bytes of the counter is incremented, where n represents the number of payload bytes in a CAIBA frame. The less-significant bytes are meanwhile reset to zero. The sender then first sends the most significant bytes of the counter for the source-authenticating tags to the authenticator. This CAN frame is protected by a regular MAC (sender and authenticator share a secret key), and the frame must only be authenticated by this one receiver. Then, the updated counter for the integrity-protecting tag is broadcasted by a CAIBA-authenticated CAN frame to all receivers. This procedure guarantees that no nonce is reused (for authentication by the sender), while even for counters that drifted apart significantly, they are reset to the same value.

5.1.5.6 Increased Reliability through Multiple Authenticators

A single authenticator is a potential weakness of CAIBA's design. Such a single point of failure can interfere with high-reliability demands of applications relying on CAN. While an inactive or misbehaving authenticator is quickly identified by the sending controllers, downgrading to unprotected communication should always be avoided.

Therefore, we propose to operate CAIBA with multiple authenticators. However, if many authenticators overwrite the same bit simultaneously, the physical signal is soon disturbed and reliable sensing is unlikely, especially due to different propagation delays. Instead, authenticators should be distributed along the bus, and only the closest authenticator takes care of overwriting a signal. Which authenticator is responsible for which ECU can be preconfigured, as ECUs are usually stationary. Such a multi-authenticator deployment has the advantage that the sender is always close to its dedicated authenticator, such that propagation delays are reduced. If an authenticator is malfunctioning, it or a noticing ECU can inform the next authenticator in line to take over. Thus, multiple authenticators increase reliability by (1) reducing the physical distance between senders and authenticators, and (2) offering fallback authenticators in case of malfunctions.

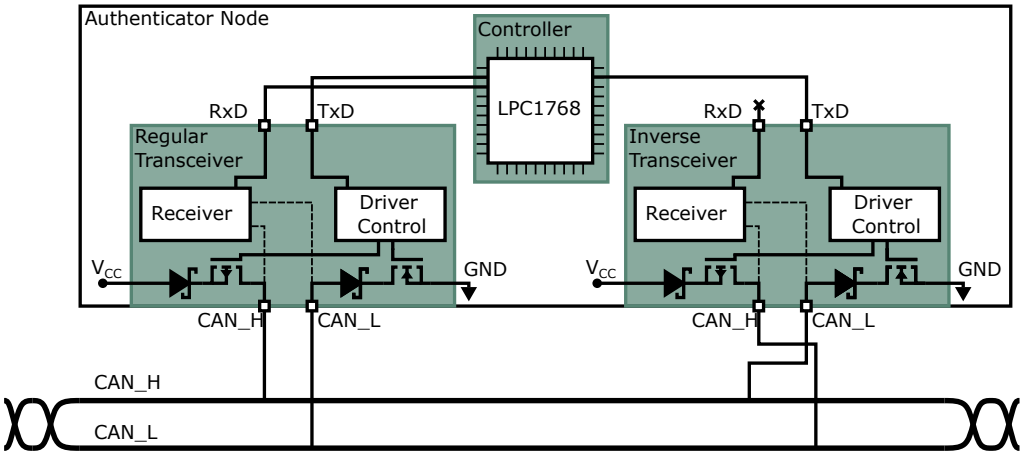


Figure 5.3 The authenticator node controls an additional CAN transceiver that is inversely connected to the bus lines to *erase* voltage differences between the bus lines.

5.1.6 CAIBA's Overwriting Mechanism

Altering CAN messages during their transmission is one key function of the authenticator in CAIBA. Technically, it requires a modification of the voltage levels on the bus lines, and it has to be timed precisely to avoid disturbing ongoing transmissions and to ensure the desired message is received by other participants. While the authenticator itself does not verify the transmitted tag t ($t = t^i \oplus t^s$), it calculates the source-authenticating tag t^s based on the payload, the nonce, and the known key k_j^{source}. By then applying the XOR operation on t and t^s (recomputed based on the received data) again within the transmission, the resulting tag t^i is read by the other devices of the bus. Recently, the feasibility of real-time perfect bit modification attacks, where the original bus state is overwritten with a static value, has been demonstrated [137]. However, neither t nor t^i is known to the authenticator in advance, such that the authenticator has to carry out these bit modification operations *reactively* and in a bit-wise manner. Specifically, the authenticator must write the inverse bit signal of what was originally written to the bus whenever a bit in t^i is set. In the following, we will describe the process of overwriting the authentication tag of a message. With RBF, we then introduce a novel technique to reactively change the physical CAN signals on-the-fly. Even though we use this technique exclusively for the CAIBA authenticator, it is worth emphasizing that RBF can be used more generally, for both defensive and offensive purposes.

5.1.6.1 Physical Signal Modification

Flipping a single bit signal requires modifying voltage levels on the bus lines. Overwriting a recessive bit with a dominant one is a core functionality of CAN to realize arbitration. Hence, the authenticator can use the normal process to write a dominant bit and thus overwrite the recessive bit of another ECU. However, if the transmitter sends a dominant bit, the authenticator must actively drive the voltages on the bus lines to the recessive state to overwrite with a recessive bit. To achieve this

overwriting, the applied voltages of the transmitter have to be reverted by sourcing CAN_L while sinking CAN_H to ground. This inverse connection, in relation to the transmitting transceiver, enables the current flow from the transceiver through the terminating resistors to be diverted off the bus, which reduces the measurable voltage drop of dominant signals. However, as CAN always operates based on the differential voltage, it is enough to keep the voltage drop below 0.5 V to make sure it is read as a recessive state [112]. Thus, additional components, which effectively erase the voltage difference, would typically need to have an internal impedance that is significantly lower than the effective impedance of both terminating resistors.

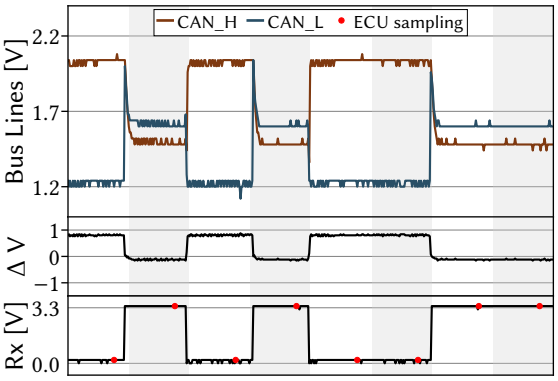
We could build such a device from discrete components. However, ordinary CAN transceivers often already offer the inverted functionality. A common internal structure of the transceiver contains two transistors to pull CAN_H to the supply voltage and CAN_L to ground [88]. We can thus connect an additional CAN transceiver with inverted bus lines to the authenticator, as shown in Figure 5.3. Then, we use the transistors to reduce the voltage difference when transmitting an apparent zero bit at this transceiver. As this would typically result in a dominant transmission, it enables a current flow through the internal transistors, which sinks the current from CAN_H to ground and sources CAN_L from the supply voltage. This changes the bus voltage to the recessive state, as can be seen in Figure 5.4b.

5.1.6.2 Reactive Bit Flipping

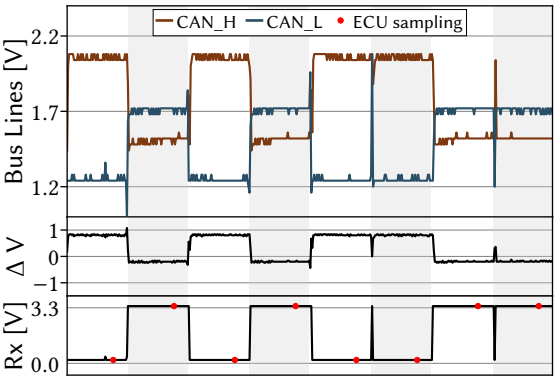
The process to flip a bit depends on the current bit of the source-authenticating tag t^s and the transmitted authentication tag t. While t^s is known, the authenticator, since not in possession of k^{group}, cannot compute a single bit of t^i before the corresponding bit of t has been transmitted by the original transmitter. However, calculating this bit after sampling the bus would result in a failing verification, since the receiver would already expect the authenticated corresponding bit of t^i. Hence, signal modifications, as described previously, require a priori knowledge about the current bit signal.

Our proposed RBF technique makes the overwriting mechanism of the authenticator *reactive*, *i.e.*, the authenticator simultaneously reads and reacts to the current state of the bus within the transmission of the same bit. To be able to read the original signal, we modify the transmission only between the third and the last time quantum of each bit signal. The first two time quanta are used to let the bus enter the state of the originally transmitted bit. At the end of the second time quantum, the authenticator samples the bus and reads the latest transmitted bit of t (*cf.* Figure 5.4c). We can expect the bus to be collision-free at this time, because it was reserved for the sender during the arbitration phase of the frame. Furthermore, we can assume that the signal has already propagated via the bus due to an approximately constant propagation delay and shift of the bit timings between ECUs. The sampled signal is then used for calculating the corresponding bit of t^i and to select the suitable CAN transceiver (*i.e.*, regular or inverse) for a possible bit alternation.

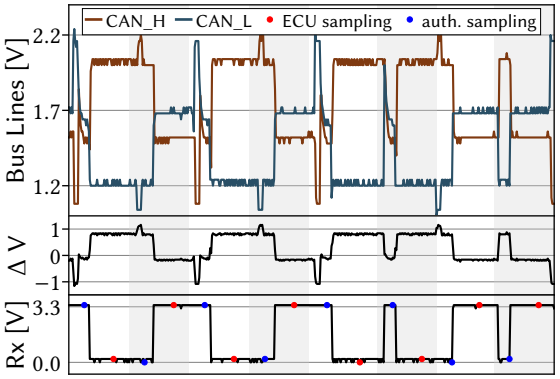
In the remaining time quanta, the authenticator forces the bus into the desired state. After 75 % of the nominal bit time has passed, ECUs sample the bus [78] and will read the authenticated bit of t^i.



(a) Unmodified transmission: 01010011.



(b) Original transmission 10101100 statically overwritten by 01010011.



(c) Reactively flipped original transmission of 10101100 to 01010011.

Figure 5.4 Showing CAN Low and CAN High for the bitstream 01010011. Figure 5.4a shown a normal transmission. In Figure 5.4b and Figure 5.4c, the original bitstream is overwritten by the authenticator with prior knowledge of the sent bits or reactively, respectively.

5.1.6.2.1 Bit Synchronization Conflict

Changing the voltage levels on the bus within a bit time can conflict with CAN's edge-oriented synchronization. Due to RBF, bits are overwritten in the third time quantum, which introduces additional edges. Concretely, a CAN controller synchronizes based on the edge when changing from a recessive state to a dominant state by prolonging the expected duration of the bit by the preconfigured Synchronization Jump Width (SJW) [110]. The SJW is 1 to 4 time quanta long and defines the maximum time by which a controller extends/shortens a bit, with a larger number generally chosen to improve robustness. However, altering the transmitted recessive bit to dominant will cause an edge in the third time quantum. Exactly if the j-th bit is read recessive ($t_j^i = 1$) and is followed by a dominant bit that is flipped ($t_{j+1} = 1$ and $t_{j+1}^i = 0$), the first detected edge occurs when the authenticator changes a signal from recessive to dominant, and all nodes resynchronize their bit time by this delayed edge. We compensate for the time shift by increasing the current bit time at the authenticator by two time quanta. The increased time the authenticator overwrites the transmission ensures that the bit value remains constant until all nodes have

sampled the bus. Furthermore, we avoid additional disturbances to the bus caused by multiple signal changes in a short period. Thus, CAIBA's authenticator compensates for the synchronization procedure embedded into all CAN ECUs.

5.1.6.2.2 Bit Stuffing Conflict

Most parts of a CAN frame are affected by bit stuffing, including the authentication tag t within the payload. Changing single bits in t can require additional stuff bits or make existing ones obsolete. Both cases could lead to an incorrect authentication tag received by other nodes or an incorrect length of the data frame and would interrupt the transmission. As stated in Section 5.1.5.2, we expect the sender to place stuff bits according to the modified bit stream that is received. To prevent overwriting stuff bits in t, the authenticator pauses the bit modification for one bit time whenever it expects a stuff bit from the sender. The position of a stuff bit is determined based on the last five regular sampling points. Although bit modification is paused, the authenticator samples the bus during the expected stuff bit to detect stuff errors or error frames.

5.1.7 Evaluation

We presented CAIBA as an innovative source authentication scheme to protect CAN without requiring significant changes to existing systems. In the following, we present a proof-of-concept implementation and show the general applicability of CAIBA. We start with the introduction of our evaluation setup and limitations. We then compare the reliability of our scheme regarding CAIBA-protected traffic. CAIBA's compatibility with legacy CAN devices is demonstrated, its processing overhead is investigated, and potential bus length restrictions is discussed. Finally, we take a look at potential adverse long-term effects on the employed hardware.

5.1.7.1 Evaluation Setup and Limitations

For our proof-of-concept implementation, we set up a testbed consisting of different ECUs as shown in Figure 5.5. To rapidly prototype the necessary modifications within the CAN controller, we used the Software-defined CAN Controller (SDCC) [52] with NXP LPC1768 microcontrollers and TI SN65HVD230 CAN transceivers. Additionally, three unmodified off-the-shelf CAN controllers were used to evaluate the backward compatibility of our solution. As the SDCC is a pure software implementation without any hardware acceleration, the maximum bitrate is limited to 40 kbit/s [52]. To show the general feasibility of CAIBA, this is sufficient.

However, to evaluate CAIBA with higher bitrates, a hardware- or FPGA-based implementation would be necessary. This is an effort we consider disproportionate to demonstrate CAIBA's general feasibility. Instead, our evaluation focuses on the practical feasibility at low bitrates, while we theoretically analyze the effects of higher bitrates *e.g.*, w.r.t. to the bus length. However, as commercial adoptations of CAIBA

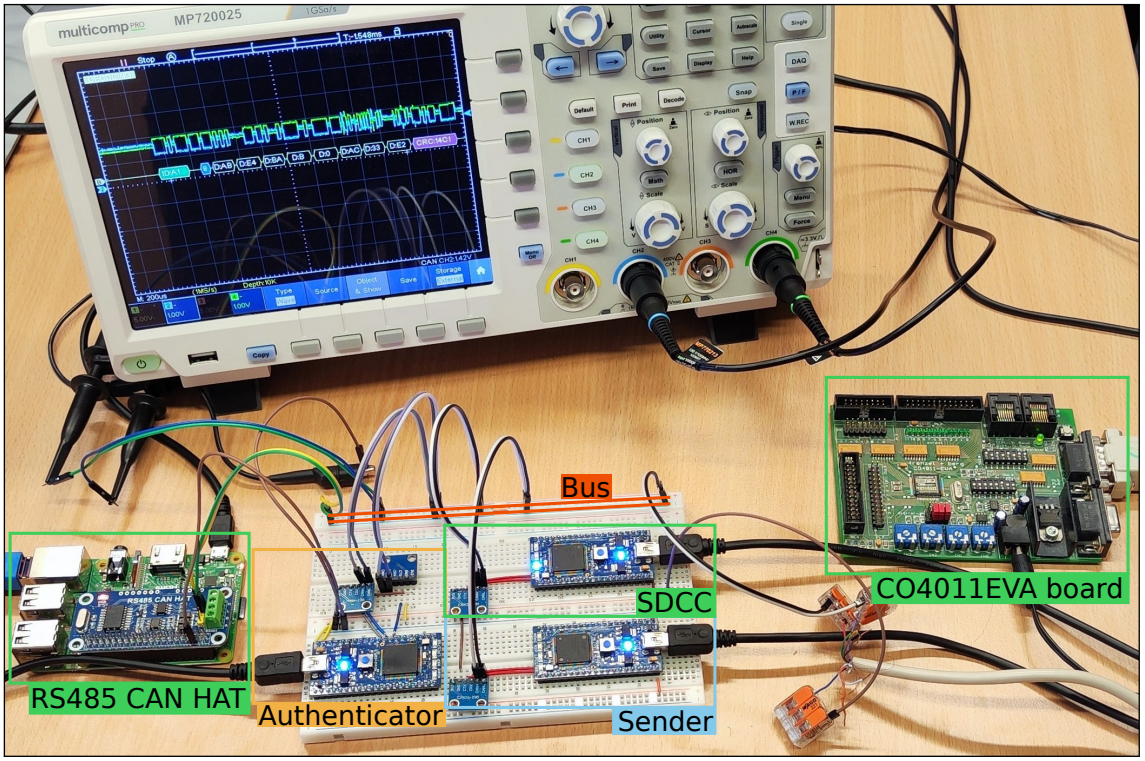


Figure 5.5 Evaluation setup of CAIBA consisting of a CAIBA-capable transmitter ●, three unmodified receivers ●, and a CAIBA authenticator ●.

are expected to be implemented in hardware, *e.g.*, as SIP cores (*cf.* Section 5.1.5.1), we do not anticipate any processing limitations even at higher speeds. For example, the most time-critical aspect is the final computation of t^s by the authenticator after the last payload bit has been received, which requires a single XOR operation of three bytes that can be computed within a single clock cycle by an FPGA.

With a faster hardware prototype, we could validate that CAIBA and legacy CAN ECUs can coexist on the same bus in an actual car. However, even then, we could not easily investigate if receiving ECUs implementing the AUTOSAR SecOC standard could be retrofitted with CAIBA's source authentication because we have no access to the secret group key used by the car's ECUs. Consequently, we have to fall back to a physical testbed for our proof of concept evaluation of CAIBA.

5.1.7.2 Reliability

Ideally, CAIBA can enable source authentication to CAN without impacting its reliability. Therefore, we compare the reliability of CAIBA to a regular SDCC-based CAN deployment. For data rates varying between 10 and 40 kbit/s, we sent 100,000 frames with 4 to 8 bytes of payload (including the authentication data) and analyze the ratio of correctly received frames at the receiver, including a valid integrity-protecting tag. To have an equal number of ECUs connected to the bus, and thus have comparable desynchronization potential, we connect an additional ECU for the CAN measurements that use no authenticator. We repeat each measurement ten times and show our results (incl. 95 % confidence intervals) in Figure 5.6.

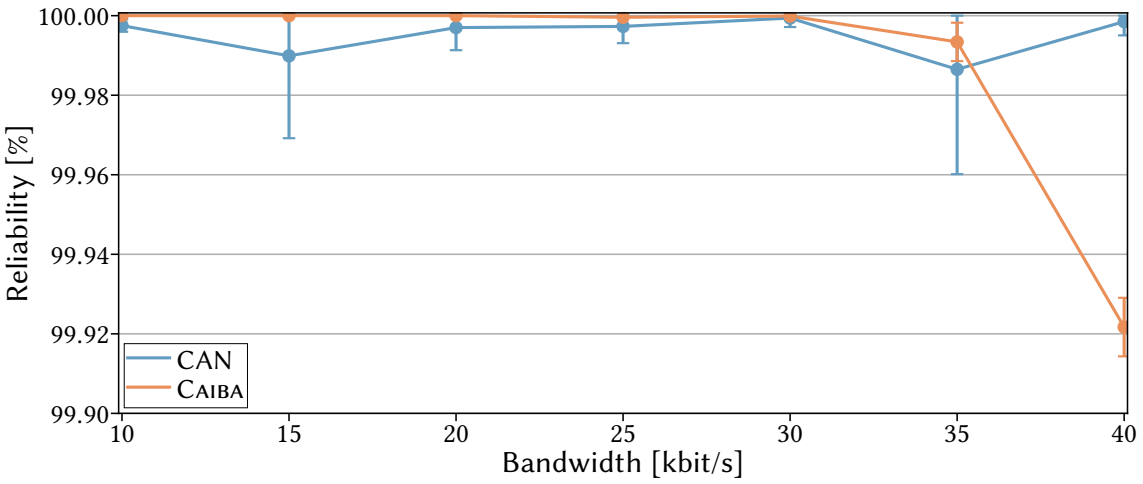


Figure 5.6 CAIBA achieves similar reliability to CAN for low data rates, but it reaches the timing limitations of the SDCC slightly earlier when increasing the rate.

For low data rates, we surprisingly observe that CAIBA achieves slightly higher reliability than CAN with no frame loss observed for data rates below 35 kbit/s. The most likely explanation for this effect is that the authenticator ECU, which acts as an additional CAN receiver for these measurements, leads to rare misinterpretations of the ACK bit. As we reach the maximum data rates supported reliably by the SDCC (*i.e.*, 40 kbit/s), CAIBA's reliability starts to decrease. Here, we reach the limitations of the timing accuracy achievable in a pure software-defined controller slightly earlier than for CAN due to the delicate bit synchronization necessary during RBF. A similar drop in reliability can also be observed in CAN when increasing the data rates further.

Overall, CAIBA can operate with similar reliability as CAN. This high reliability can, however, only be achieved with an authenticator providing precise timing relative to the bus speed. For higher data rates, this is only achievable through hardware implementations of CAIBA controllers.

5.1.7.3 Compatibility with Legacy CAN Devices

We cannot expect that each device on a bus is aware of CAIBA. Otherwise, car manufacturers would have to convince each ECU supplier to adopt CAIBA, before the first car can employ multicast source authentication. Thus, CAIBA is specifically designed not to interfere with legacy communication. Also, only transmitters have to be altered to support CAIBA, while AUTOSAR SecOC-implementing receivers can still benefit from the provided security without any modifications (*cf.* Section 5.1.5).

To validate the compatibility with legacy devices, we tested CAIBA in combination with three additional off-the-shelf receivers. In particular, we used CAIBA in combination with an MCP2515 CAN controller connected to a Raspberry Pi 3 trough SPI in the form of the *Waveshare RS485 CAN HAT* [254], the integrated CAN interpreter of a *MULTICOMP PRO MP720025 EU-UK* [167] oscilloscope, and a CANOpen Evaluation Board using the *Frenzel+Berg CO4011A Controller* [81]. All

three devices served as regular receivers for authenticated CAN messages that were modified by the CAIBA authenticator. As a result, no anomalies have been observed when reading CAN messages with any device. Thus, we conclude that CAIBA is indeed compatible with unmodified receivers and legacy CAN communication.

5.1.7.4 Upper Bound on Processing Overhead

In the literature, cryptographic approaches to protect the CAN bus are often criticized for their excessive processing overhead. Indeed, the computation of a single HMAC-SHA256 tag on an ARM Cortex M3 (32 MHz) takes $1641.4 \pm 0.1\mu s$. The ARM Cortex M3 is a typical prototyping processor with similar processing power to commercial ECUs, which are, however, built to operate reliably even in harsh environments. The computation of one tag thus takes over 10 times longer than the transmission of a single CAN frame at 1 Mbit/s (128 μs). In other words, a corresponding ECU could only verify or authenticate one tenth of all CAN frames transmitted on a fully utilized 1 Mbit/s bus. To assess the overhead of CAIBA on the ECUs, we measured the duration to compute a tag (consisting of t^s and t^i) with a length of 24 bit for payload lengths varying between 1 and 5 bytes.

For the integrity-protecting tag t^i, we compute a CMAC as recommended in AUTOSAR SecOC. On the other hand, the source-authenticating tag t^s is computed based on BP-MAC for fast online computation by the authenticator. As our results in Figure 5.7 demonstrate, the overhead of the additional tag computation is at most 51 % in total. Here, a fixed overhead of 19.9 μs is caused by the preprocessed computation of blinding tags. The actual additional delay during tag computation only amounts to between 14.6 μs and 31.3 μs, depending on the payload length. Moreover, we observe that if AUTOSAR SecOC were to adopt BP-MAC for its tag computation, it could reduce its processing time by 69.0 μs, *i.e.*, CAIBA would then be faster than AUTOSAR SecOC with its recommended MAC scheme of today.

As we expect that commercially deployed CAIBA controllers are implemented in hardware, the overhead of BP-MAC is suspected to further reduce significantly [250]. Overall, we found that cryptographic processing is not a significant drawback for ECUs, even for cryptographic processing in software, due to the selection of a fast, yet secure, MAC scheme.

5.1.7.5 No Limitations to the Bus Length

As the authenticator's overwritten signal must be reliably sampled by the receiving ECUs, we now look at potential bus length restrictions. In the worst case, the originally transmitted signal and the overwritten signal are delayed by twice the distance between the sender and the authenticator. This maximal offset occurs if the receiver is placed near the sender on one extremity of the bus with a maximum distance to the centrally placed authenticator. Hence, one might expect a reduction in the maximum CAN cable length by the employment of CAIBA. However, adding up the delays for signal propagation, overwriting, and synchronization, even for the

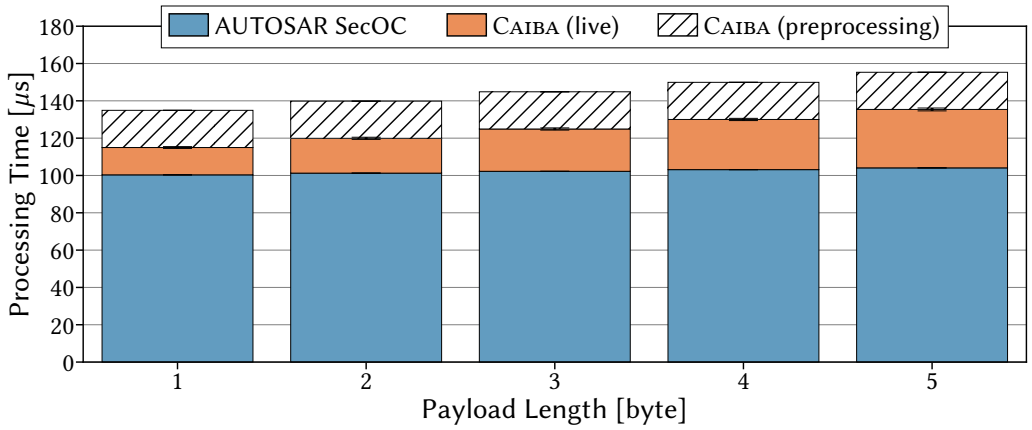


Figure 5.7 Processing overhead of CAIBA's tag computation on a representative ARM Cortex M3 chip is significantly lower than a single frame transmission (128 μs) even if CAN is operated at 1 Mbit/s.

maximal bus lengths, CAIBA's delays are still tolerable according to the acceptable intervals for all bit rates of CAN (assuming a single centrally placed authenticator).

Looking at the example for 1 Mbit/s, CAN supports a maximum cable length of 25 m and a receiver sampling point 750 ns after the start of a bit. Adding the propagation delay of traveling halfway and back (125 ns), three quanta idle time for synchronization and back-propagation (375 ns), and the authenticator's transceiver delay (210 ns) results in a worst-case delay of 690 ns[3]. Hence, even with a maximum CAN cable length and worst-case sender and receiver placement, the overwritten signal is still stable at the receiver before it samples the bus 750 ns after the start of the signal. Using CAIBA thus does not restrict maximum cable length, as other aspects of CAN are more restrictive, *e.g.*, arbitration and ACK bits that must function over the entire length of the bus. Meanwhile, the centrally placed authenticator is closer to the sender and thus operates with lower propagation delay.

To practically validate that CAIBA can operate on longer buses, we placed a 50 m twisted pair cable between the authenticator on one end of the cable, and the sending as well as the receiving ECU on the other end of the cable. This corresponds to the worst-case scenario for a 100 m long bus. We transmit 10,000 CAN frames at 40 kbit/s and repeat this measurement ten times. Note that at these speeds CAN could operate on a bus 10 times as long, but only reliably with the use of optocouplers [78]. We achieve a reliability of $99.95 \pm 0.03\,\%$. While this reliability is even slightly higher than for the short bus ($99.92 \pm 0.01\,\%$), potentially due to better stabilizing of the signal during propagation, they lie within the margin of error of each other. Overall, we can thus conclude that CAIBA can operate reliably even on longer buses and does, in theory, not restrict the maximum bus length at all.

[3] Baseline numbers stem from the CANopen standard [78].

5.1.7.6 Long-Term Impact of Overwriting Bits

Finally, we study potential long-term adverse effects of CAIBA on the hardware. When the authenticator overwrites a dominant bit, the inversely connected transceiver sinks the current from the bus line, resulting in an actively driven recessive state. While this can be seen as a short circuit on the bus (from the view of the original transceiver) with a dampened current, all transceivers must be short-circuit proof for fault tolerance [108].

Additionally, we measure the current that flows through the affected transceivers during the overwriting of dominant bits. We observe a maximum current of 28 mA, which is within specifications for common CAN transceivers such as the Philips TCA1050 (100 mA) [191] or the TI SN65HVD25x (160 mA) [234]. Most importantly, these currents only affect CAIBA transceivers. Legacy ECUs connected to the same bus are not affected as they only listen when dominant bits are overwritten. Consequently, there exists no risk for long-term effects of operating CAIBA in CAN.

5.1.8 Limitations and Future Challenges

We present CAIBA as a novel solution to the multicast source authentication problem and thus protect CAN. In this context, we provide a proof-of-concept implementation and show its compatibility and dependence in a range of evaluations. In the following, we identify current limitations and potential future improvements for CAIBA.

Broader Applicability of CAIBA. CAIBA promises multicast source authentication without verification delay or excessive bandwidth requirements. Indeed, we demonstrate CAIBA's seamless integration into the CAN bus protocol. However, it remains to be investigated how applicable CAIBA is beyond CAN. Other automotive buses, such as LIN [111] and FlexRay [109], show great potential but also pose some unique challenges. The potential of CAIBA in these and other networks, such as star topologies with the central switch taking on the role of the authenticator, still needs to be explored.

Hardware-based CAIBA ECUs. We prototypically show the applicability of CAIBA on a software-defined CAN controller and its interoperability with off-the-shelf CAN controllers. However, inherent performance limitations and timing inaccuracies of software-defined controllers limit the bus speeds. A hardware-based CAIBA controller is thus needed for the deployment of CAIBA with typical bus speeds in cars.

CAIBA alongside Intrusion Detection. Previously, we argued that we should not rely solely on IDS because of their imperfect detection, false alarms, and potential evasions. Nonetheless, IDSs do not become superfluous because of CAIBA. In contrast, the operation of CAIBA produces characteristic behavior that can even facilitate monitoring by an adapted intrusion detection mechanism to detect any anomalous behavior early. Such potential symbiosis should be further investigated.

Hardening the Authenticator Module. Thus far, we assume a trusted authenticator that is hard to compromise, even with physical access to the CAN bus. This

assumption is in line with other proposals, *e.g.*, IDSs for in-vehicular communication. Still, the concrete steps to maximize temper-resilience beyond offering no external connectivity must be worked out.

5.1.9 Summary

The increasing connectivity of modern vehicles, coupled with the vulnerability of the CAN bus, has raised significant concerns regarding the safety and security of automotive systems. Compromised ECUs of, *e.g.*, infotainment systems, can easily masquerade as other entities in the network and take over control of a vehicle. Current security mechanisms mostly rely on intrusion detection or group key-based authentication tags, with both approaches not protecting sufficiently against masquerading attacks. In contrast, we propose a novel multicast source authentication scheme (CAIBA) for buses that, unlike prior approaches, does not require longer tags or delayed verification. CAIBA relies on an authenticator that reactively overwrites bits of the authentication tag included in each message. The keys to generate the original tags are only known by the genuine source, while all receivers can verify the altered tags. We show the applicability of CAIBA in CAN, providing source authentication for ECUs implementing the AUTOSAR SecOC specification. CAIBA is incrementally deployable and interoperable with legacy CAN traffic, while achieving high reliability with minimal processing overhead.

5.2 Middlebox-aware End-to-end Secured Channels

In Section 5.1, we explore the potential of MACs to achieve efficient multicast source authentication. In this section, we now want to explore another direction to protect communication between more than two parties. This time, we are, however, not interested in incorporating multiple receivers, but in selectively giving read and write access to middleboxes encountered during routing. Concretely, we introduce *Middlebox-Aware DTLS* (MADTLS). MADTLS provides fine-grained access control to middleboxes via specialized cryptographic protocols. We enable the transparent segmentation of messages to assign read and write access on a per-segment granularity. Therefore, each segment is encrypted and authenticated individually. Here, MADTLS leverages a specifically tailored message authentication scheme to aggregate authentication data, thus conserving valuable bandwidth. Moreover, MADTLS provides an extension to the DTLS 1.2 handshake protocol to exchange the additional information required by the communicating endpoints and middleboxes. While industrial networks may rely on reliable (*e.g.*, TCP) or lossy channels (*e.g.*, UDP, with a recent focus on wireless communication), we specifically target the more challenging domain where packets may be lost arbitrarily. Still, the presented techniques are transferable to TLS (over TCP) and other end-to-end protocols.

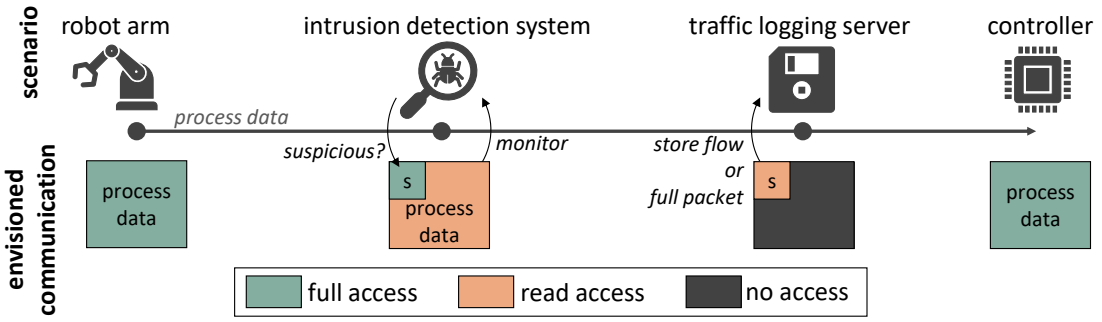


Figure 5.8 Middleboxes should operate in a least-privilege mode. For example, an industrial IDS has read-only access to packets and write access to a dedicated flag to mark suspicious traffic. A logging server can only read this flag.

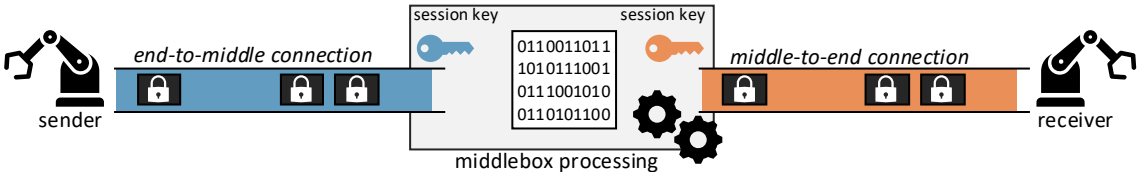
5.2.1 Motivation

While TLS experiences wide-scale adoption in the traditional Internet, the prospects of deployments in industrial networks look rather grim. One major roadblock in deploying end-to-end security is the widespread use of middleboxes that rely on deep packet inspection: Traditional middlebox functionality, such as intrusion detection, requires access to sensor and actuator data to monitor industrial processes [260]; Proposed process-aware IDS for industrial networks *e.g.*, validate the plausability of process data [258] or predict the physical state based on previous observations [259]. Likewise, the increasing interest in in-network computing likewise requires read and write access to in-flight data [158]. Hence, middleboxes prevent the deployment of traditional end-to-end security, raising a serious security dilemma.

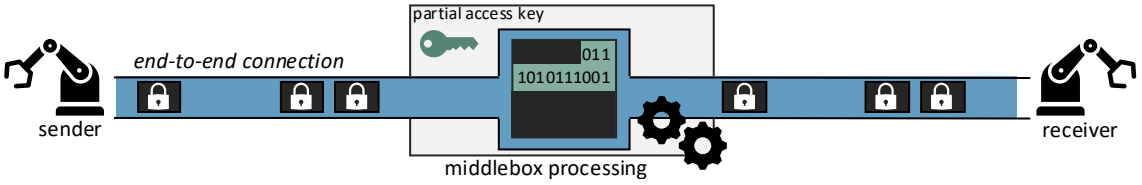
To depict this issue, we consider an illustrative example of how interconnected Industrial IoT (IIoT) communication may look in the future in Figure 5.8. A controller operates a robot arm. An IDS and a traffic logging server monitor their communication. The industrial IDS analyzes traffic to detect anomalies and flags suspicious packets. The traffic logging server only captures flow metadata but stores suspicious packets entirely for potential subsequent forensic investigations.

The naïve approach for middlebox access control while still providing security is to create separate secure point-to-point connections, one from robot arm to IDS, one from IDS to logging server, and a final one from logging server to controller. This approach, commonly referred to as SplitTLS [176], gives each middlebox full access to manipulate messages. Consequently, a compromised IDS could move the robot arm unexpectedly and thus damage expensive equipment or cause physical harm. Since middleboxes are typically placed at critical vantage points with access to large amounts of traffic, they then become especially attractive targets for attackers. Therefore, it is of utmost importance to deploy middleboxes in a *least-privilege mode*. Considering the IDS from our example, it should only have read access, except for writing to a single flag to mark suspicious packets. In contrast, SplitTLS provides full insight and control over the entire communication channel to middleboxes.

To address these limitations, *middlebox-awareness* has been proposed to adapt end-to-end security protocols to the reality of middleboxes in corporate networks and



(a) SplitTLS solutions allow full access to middleboxes.



(b) A least-privilege access for middleboxes would be preferable.

Figure 5.9 While SplitTLS allows full access to middleboxes, achieving least-privilege access for middleboxes should be the goal for middlebox-aware security protocols.

the Internet [63]. A first branch of research relies on searchable encryption [43, 134, 181, 223] or zero-knowledge proofs [93, 269] to perform a limited set of computations (*e.g.*, string matching) on encrypted data. While these approaches may work for basic rule-based intrusion detection, they are too restrictive for most middleboxes in the industrial context. Other proposals move middleboxes into Trusted Execution Environments (TEEs) [68, 98, 192, 237]. While such approaches work great in theory, the secure implementation of the concept of TEEs is difficult in practice, as recent attacks have shown [40, 169]. Finally, a branch of research extends TLS to allow for the authorization of on-path middleboxes to read or write to a predetermined set of communicated data [12, 76, 136, 142, 175, 176]. Still, operating on the granularity of complete messages or only supporting two access modes, these proposals do not provide the fine-grained access control necessary to operate middleboxes in a least-privilege mode.

While middlebox-aware end-to-end security thus offers an attractive solution for the security dilemma, current proposals do not address the unique challenges of industrial communication. First, tight latency and bandwidth demands, paired with resource-constrained embedded hardware, require fast and efficient protocols [155]. Secondly, predictable and well-structured messages enable, but also require, fine-grained control down to the *bit level* (instead of complete messages) over the access rights of each involved middlebox to constrain their privileges to a minimum. Thereby, we can minimize the damage inflicted by potentially compromised middleboxes. Thirdly, middleboxes may need to inject entirely new messages (*e.g.*, emergency shutdowns) into established communication channels, where the least-privilege principle must also be upheld.

5.2.2 Middleboxes in the Industrial IoT

Middleboxes play an integral role in the industrial IoT for performance enhancements and intrusion detection [145], but they also severely hinder the adoption of

traditional end-to-end security. Addressing this lack of security and also accounting for the high demands in industrial networks, we observe an increasing interest in middlebox deployments [158]. Security-wise, these middleboxes can, *e.g.*, realize deep packet inspection for intrusion detection. Regarding performance, the full range of functionality extends to a diverse set of tasks including caching [95, 143], data aggregation [212], fault detection [213], and in-network processing [53, 205, 209].

While these advancements alleviate specific limitations and *some* security flaws within the IIoT, they also hinder properly deploying end-to-end security protocols, as we demonstrate in Figure 5.9. Current industrial deployments of end-to-end security are mainly realized with TLS or DTLS and can, at most, provide a so-called SplitTLS [176] solution, where middleboxes can freely access and modify any traffic that passes through them (*cf.* Figure 5.9a). However, middleboxes are usually deployed for a particular task, so an ideal security protocol would restrict read and write access to a minimum (*cf.* Figure 5.9b). Only then can the IIoT operate in a least-privilege mode and minimize the damage of compromised middleboxes.

5.2.2.1 Diversity of Industrial Middlebox Use Cases

To understand the requirements for middlebox-aware end-to-end security in industrial networks, we need to understand the range of potential tasks. Therefore, we categorize corresponding middlebox applications according to their required access rights.

Read Access. Examples for read access include a middlebox that caches, *e.g.*, sensor readings to unburden the constrained end devices [95, 143]. Similarly, industrial IDSs monitoring the physical process to detect suspicious activities only requires read access to otherwise encrypted data for deep packet inspection [260].

Limited Read Access. However, even for tasks such as intrusion detection or caching, middleboxes often only require partial insight into each message. For example, most Snort [6] rulesets for Modbus (a widespread industrial communication protocol) do not look beyond the function code field (indicating the type of a message, *e.g.*, request the state of an individual bit). Even more sophisticated industrial IDSs only require partial read access to messages in some cases [48, 49, 146]. Moreover, other use cases only require partial read access by design: For fault detection, middleboxes only need access to selected sensor readings from specific machinery to detect upcoming failures [213]. Similar access rights to only specific sensor readings are necessary for complex event detection, *e.g.*, the outbreak of a fire [121, 158, 242].

Write Access. Middlebox tasks that require *full* write access to messages are rare, *e.g.*, when the middlebox translates between different application layer protocols [2, 239]. Instead, most middleboxes that alter messages only require limited write access.

Limited Write Access. A typical example of limited write access in the industrial context and beyond is data compression, where a middlebox can, *e.g.*, base-delta encode timestamps for many different data sources [186]. Similarly, in-network aggregation or other map-reduce functionality can be executed by middleboxes that only have write access to the corresponding data fields they are reducing [212]. More industry-specific applications for restricted write access include, *e.g.*, the

transformation of coordinates between reference frames [129] or the insertion of precise timestamps into payload data [128]. Furthermore, as seen in the example in Section 5.2, various middleboxes may take advantage of flagging individual packets, *e.g.*, to mark them as suspicious.

Drop Messages. In the industrial context, it is often not desirable to directly drop messages due to irrevocable impact on the physical process controlled by the system. Thus, industrial networks mostly only employ IDSs that flag suspicious traffic or alert the operator through other channels. Still, for some use cases, a middlebox requires the ability to drop messages even in industrial networks, *e.g.*, to downsample sensor readings [130, 242], carefully reduce traffic [96], or if a serious cyberattack is identified with high likelihood (*i.e.*, the benefit of likely preventing a high-impact attack outweighs the risk of blocking genuine traffic).

Inject Messages. Finally, middleboxes may also require the ability to inject messages, most importantly when a middlebox directly responds to a request or event to reduce latency or traffic. Complex event detection could, *e.g.*, identify critical conditions such as a fire [121], that warrant the issuing of emergency stop messages [53]. Other reasons to allow middleboxes to inject messages include responding caching servers [95, 143] or the enabling of low-latency and low-jitter control commands [53, 205, 209].

Overall, we see that middleboxes in the IIoT cover a diverse set of functionalities that require different levels of access to a communication channel. Most importantly, these functionalities are often specific to the IIoT and must be considered when designing middlebox-aware end-to-end security.

5.2.2.2 Prior Work on Middlebox-aware Security

Research on such *middlebox-aware security* protocols goes back over a decade for the traditional Internet [63], but we still see few deployments today. Corresponding concerns can, among other things, be traced back to the Internet community's end-to-end principle [45, 46], which is typically interpreted to imply that the functionality in the network should be kept minimal while end-hosts implement most, if not all, functionality. Thus, especially in security contexts, the addition of middleboxes terminating end-to-end connections is often regarded as a slippery slope toward security and privacy loss on the Internet. In contrast to the general Internet, limited domains [47] allow for more liberal solutions that can be tailored to the needs of the domain. Industrial networks are one prominent example of such a limited domain, as they are typically under a single administrative control. Additionally, all devices follow a common goal: ensuring the successful operation of the industrial process. Industrial networks and other limited domains thus represent a contrasting deployment scenario compared to the Internet, calling to revisit secure middlebox-aware communication specifically from the perspective of industrial communication. Since current proposals for middlebox-aware end-to-end security protocols are designed for the general Internet, we now investigate to what extent existing proposals are suited for industrial deployments. Table 5.2 summarizes the results of our analysis.

An initial set of proposals attempts to realize middlebox functionality directly on encrypted traffic [43, 134, 181, 223]. Here, BlindBox [223] introduces the original idea

	Publication	Venue	read full	read lim.	write full	write lim.	inj.	drop	diff.
search. enc.	BlindBox [223]	SIGCOMM	◐	◐	○	○	○	●	○
	Embark [134]	NSDI	◐	◐	○	○	○	●	◐
	BlindIDS [43]	AsiaCCS	◐	◐	○	○	○	●	○
	PrivDPI [181]	CCS	◐	◐	○	○	○	●	○
ZKPs	ZKMB [93]	USENIX Sec.	◐	◐	○	○	○	◐	●
	Zombie [269]	IEEE S&P	◐	◐	○	○	○	◐	●
TEEs	SGX-Box [98]	APNet	●	○	●	○	○	◐	○
	Safebricks [192]	NSDI	●	◐	●	◐	○	◐	●
	Shieldbox [237]	SOSR	●	○	●	○	○	◐	○
	Lightbox [68]	CCS	●	○	●	○	○	◐	○
protocol extension	mcTLS [176]	SIGCOMM	●	◐	●	◐	○	○	●
	mbTLS [175]	CoNEXT	○	○	●	○	○	○	●
	maTLS [136]	NDSS	●	◐	●	◐	○	○	●
	ME-TLS [142]	IEEE IoTJ	●	◐	●	◐	○	○	●
	Stealth Key Exchange [76]	CCS	●	◐	●	◐	○	○	◐
	mdTLS [12]	ICISC	●	◐	●	◐	○	○	●
	MADTLS [247]	AsiaCCS	●	●	●	●	●	◐	●

●: yes ◐: partial ○: no

Table 5.2 The current state-of-the-art on middlebox-aware security protocols cannot address all requirements of industrial networks. Searchable encryption or zero-knowledge proofs only provide limited functionality (*e.g.*, string matching). Vulnerabilities within TEEs expose all approaches relying on them. Finally, extensions to the TLS protocol do not provide the fine-grained access control required to truly operate middleboxes in a least-privilege mode. Moreover, no proposal offers features to give middleboxes the ability to inject (a restricted set of) messages into the communication stream.

but only enables the functionality to evaluate regular expressions on encrypted data. Subsequent efforts extend this functionality [134], improve performance by reusing computations [181], or focus on specific use cases such as intrusion detection [43]. However, searchable encryption enables only a limited set of computations on network traffic, does not support altering or injecting traffic, and brings unacceptable performance penalties to resource-constrained industrial devices. A similar picture is drawn by the recent first proposal to employ zero-knowledge proofs provided by clients that attest the abidance to certain rules, *e.g.*, to prevent routing traffic to a blacklisted IP address [93]. Again, this approach has significant performance drawbacks, restricted functionality, and no support for altering or injecting traffic.

A fundamentally different idea is to encapsulate middlebox functionality into TEEs to shield them against malicious or compromised hosts [68, 98, 192, 237]. These

approaches, such as SGX-Box [98] share TLS session keys with the TEE. Within the TEE, packets are entirely decrypted and can even be altered by the middlebox. Lightbox [68] further protects traffic metadata and enables stateful middleboxes. Shieldbox [237] facilitates the implementation of middleboxes in Intel SGX via the Click framework [120]. Meanwhile, SafeBricks [192] restricts middleboxes to only partially access packets and improves the performance of chained middleboxes. However, the access restriction of SafeBricks does not cryptographically protect against malicious middleboxes and relies on a correct realization of Rust's type system and the security of the TEE itself. Still, the limited memory of TEEs impedes the application of these proposals even if TEEs were available in industrial settings. Furthermore, all protocols are designed for TLS, such that dropping individual messages implies that all future sequence numbers must be adapted. More importantly, all these approaches assume a secure implementation of the TEE primitive, which a plethora of recent attacks (*e.g.*, Plundervolt [169] or Æpic leak [40]) have shown to be difficult.

Finally, recent work investigates the direct integration of middleboxes into TLS sessions via protocol extensions [12, 76, 136, 142, 175, 176]. In mcTLS [176], each message is assigned a preconfigured context consisting of a unique set of middleboxes with read and/or write access. The TLS record protocol header is then extended by a context identifier and two authentication tags, one for readers and one for writers. Readers verify the reader tags, writers verify reader and writer tags, and endpoints verify all tags. Thus, readers can verify that no third party modified the packet, writers know that all modifications stem from writers, and endpoints additionally know whether the packet has been modified at all. Still, mcTLS only provides coarse access control on a per-message level. As an alternative to mcTLS, maTLS [136] proposes to use different TLS sessions for middleboxes and append a modification log to each packet to track changes. Despite performance improvements in mdTLS [12], this approach introduces significant processing delay and bandwidth overhead. In turn, ME-TLS [142] improves the handshake efficiency over that of mcTLS by providing authorized middleboxes with the necessary information to recover session keys from passively observed handshake messages. Furthermore, mbTLS [175] enables the dynamic integration of middleboxes into a special TLS session where a new key is used for each middlebox to enforce path integrity, but without restricting data access. Stealth key exchanges [76] take yet another route by exchanging a secondary encryption key in standard-conform TLS 1.3 communication that can be used to keep traffic encrypted and/or authenticated when sharing the primary session key with a middlebox.

Overall, even these closely related protocol proposals only provide access control on a per-message basis and often introduce significant overhead. In general, the current state-of-the-art on middlebox-aware end-to-end security protocols cannot provide the fine-grained access control to packets necessary to operate middleboxes in a least-privilege mode. Furthermore, no current proposal covers the case of middleboxes injecting traffic, as may be necessary within industrial applications to, *e.g.*, issue emergency stop commands [53] or reduce the latency of control tasks [53,205, 209]. Consequently, we conclude that current middlebox-aware end-to-end security protocols are not suited for industrial networks.

5.2.3 Threat Model & Requirements

Despite extensive research on middlebox-aware end-to-end security, industrial networks can still not be efficiently equipped with such functionality. Meanwhile, these networks offer a prime target for such approaches, as established middlebox deployments often prevent other security measures. In the following, we first establish the threat model against which middlebox-aware end-to-end security (in industrial networks) must protect. Then, we distill concrete requirements that must be fulfilled to enable the least-privilege operation of middleboxes.

5.2.3.1 Threat Model

We strive to prevent attacks aiming at unauthorized read or write access to communication channels in industrial networks. To achieve this goal, we consider an attacker according to the Dolev-Yao threat model [67], *i.e.*, an attacker which has complete control over the entire network (but neither of the two endpoints of a communication channel). Accordingly, the attacker can arbitrarily read, alter, reroute, inject, and drop packets. Additionally, an attacker may arbitrarily compromise one or multiple middleboxes that access the communication channel. Within our threat model, the attack must be constrained from extending its control over a communication session beyond the minimum access requirements any compromised middleboxes have over that session. Furthermore, an attack must be prevented from altering middleboxes' order or skipping some middleboxes entirely, as this may lead to incorrect data being processed or ineffective intrusion detection. Denial of Service (DoS) attacks and side-channel attacks are out of scope of MADTLS's design as these kinds of attacks affect all security protocols.

5.2.3.2 Requirements

Middlebox-aware security protocols should provide the same security guarantees as end-to-end protocols towards outsiders, *i.e.*, entities not part of the communication. Concretely, any communication session should authenticate the communication endpoints, provide data secrecy, and ensure data integrity. Beyond these inherited requirements, middlebox-aware end-to-end security protocols (for industrial networks) must fulfill additional requirements regarding the integration of middleboxes into communication sessions.

Explicit Middlebox Authentication. The authentication of endpoints in end-to-end security must be extended to all middleboxes with read or write access to any message. Thus, both endpoints must explicitly acknowledge and verify all middleboxes involved in the communication and the privileges they have been assigned.

Least Privilege Read and Write Access. Middleboxes are often located at critical vantage points to process as much traffic as possible. This makes middleboxes a particularly attractive target for attacks, while they often fulfill dedicated tasks that require read and/or write access to specific parts of messages. Consequently,

middleboxes should operate in a least-privilege mode where they are restricted to exactly those access rights that are inevitable to fulfill their task. The selection of applications from Section 5.2.2.1 shows that this access might have to be restricted to bit-wise read and/or write access to specific fields in a message.

Least Privilege Traffic Injection. Some middlebox tasks need to inject control commands for low-latency control or emergency shutdowns (*cf.* Section 5.2.2.1). However, such capabilities must not lead to full control over the communication channels, *i.e.*, a middlebox should not be able to inject arbitrary traffic. Instead, privileges to inject traffic must be restricted to the minimum required for correct functionality, *e.g.*, to only inject messages with specific Modbus function codes.

Path Integrity. Generally, the order in which middleboxes process a message is important. A middlebox performing complex computation may, *e.g.*, need to be placed behind a filtering middlebox to be able to keep up at line-rate. Therefore, an attacker should not be able to change the processing order of middleboxes, or worse, skip certain middleboxes (*e.g.*, an IDS) entirely. To prevent such attacks, a middlebox-aware security protocol should enforce path integrity, *i.e.*, a message's correct verification depends on it passing all intended middleboxes in the right order.

Accounting for Resource Constraints. Middlebox-aware end-to-end security for industrial networks has to account for resource-constrained devices and networks. More specifically, adequate latencies must be ensured even with limited processing power. Additionally, any per-message overhead for short messages should be minimized to conserve bandwidth.

Our requirements show similarities with the traditional Internet (*e.g.*, path integrity [175]), but also a range of challenges unique to the IIoT (*e.g.*, least-privilege traffic injection). As the current state-of-the-art cannot fulfill all these requirements, we propose a middlebox-aware security protocol tailored to the IIoT.

5.2.4 High-level Design of MADTLS

To enable middlebox-aware end-to-end security in industrial networks, we propose Middlebox-Aware DTLS (short: MADTLS). We choose to integrate MADTLS as an extension to DTLS as it represents the bigger challenge compared to TLS, since any message can be lost during its transmission. However, the integration into, *e.g.*, TLS 1.3 should be possible without significant changes. MADTLS is designed to fulfill the requirements for industrial networks outlined in the previous section. Still, MADTLS could also be advantageous in other scenarios such as data center networks.

The main idea underlying MADTLS is to partition messages into segments, which are then individually encrypted and authenticated such that middleboxes can only read or write to a specific subset of those segments. We illustrate the entire process of securing a message before transmission, over the partial access by a middlebox, until the final reception of the message at its destination in Figure 5.10. First, ① the sender partitions a message into non-overlapping segments such that a specific set of access rights for each middlebox can be assigned to each segment. The resulting

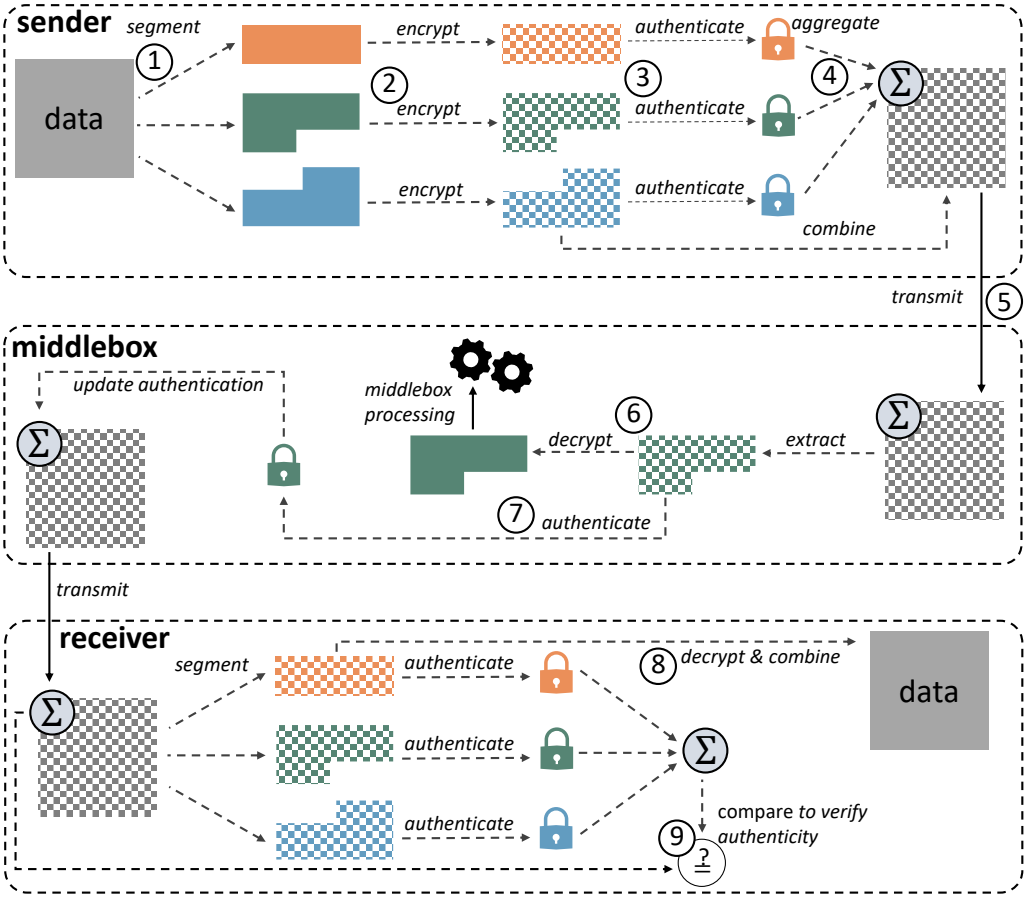


Figure 5.10 The core idea behind Madtls is to segment messages according to predetermined templates, which allows us to encrypt and authenticate each segment individually. To save bandwidth, segment authentication is only verified by the receiver and selected middleboxes. Therefore, each middlebox updates the authentication tag, such that later verification of this tag ensures path integrity and data integrity at all times.

segments are then ② separately encrypted (*e.g.*, with AES in counter mode), and ③ an authentication tag is computed for each segment. Notably, the authentication scheme for individual segments is designed such that ④ all authentication tags for individual segments can be aggregated to save valuable bandwidth. The encrypted segments and the aggregated authentication tag are then ⑤ transmitted to their destination and intercepted by a middlebox.

In Figure 5.10, the middlebox has read access to the segment in the middle (green). After ⑥ decrypting this segment, it can process the contained data to realize its middlebox functionality. Meanwhile, the middlebox also ⑦ updates the aggregated authentication tag such that the final receiver can retrace whether this middlebox received the correct data. Thus, the receiver can verify that no attacker manipulated data before a middlebox processes it and reverted these changes after the middlebox processed it.

The final receiver ⑧ decrypts all segments and ⑨ computes an authentication tag over the received data to compare it to the transmitted tag. If both tags match, the integrity of the transmitted data is proven: The receiver *and all middleboxes* received

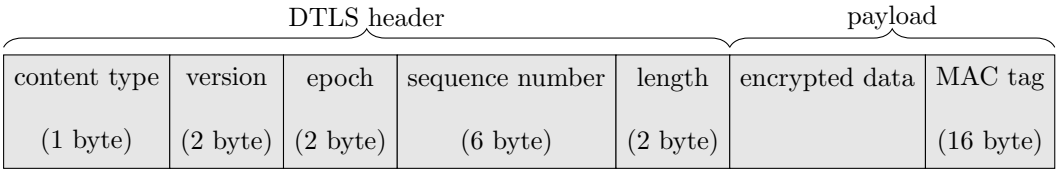


Figure 5.11 The DTLS 1.2 header format realizes a concise message format that can easily be extended to support new functionality through the addition of content types.

data that has only been modified by middleboxes that have been explicitly allowed to make these changes. After discussing MADTLS on a high level, we now dive into the details of the record layer protocol.

5.2.5 The MADTLS Record Protocol

To introduce the idea of middlebox-awareness to industrial networks, we extend the DTLS 1.2 protocol. We start with a summary of the DTLS 1.2 record layer layouts. Then, we discuss how we extend this layout to support limited read and write access for middleboxes. We then discuss how encryption and integrity protection are handled. Finally, we discuss extensions to realize integrity verification by middleboxes and limited data injection.

5.2.5.1 Recap: DTLS Record Layer

We first describe the DTLS record layer in Figure 5.11. The `content type` is used to identify the type of messages (*e.g.*, DTLS handshake messages are represented by the type `0x16`). New content types can be added to extend the functionality of DTLS. The `version` field indicates the protocol version (*e.g.*, 1.2). Then, DTLS uses two fields that combine to form a nonce to prevent replay attacks. The `epoch` tracks cipher suit changes, and a zero-initialized `sequence number` increments with each message. The `length` field then indicates the length of the DTLS record layer payload, which is appended afterward. To this end, the payload is encrypted, and an authenticated tag is appended according to the chosen cipher suite. In summary, DTLS realizes a concise message format that can easily be extended to new functionality through new content types.

5.2.5.2 MADTLS' Record Layer Header Structure

To realize MADTLS, we extend the DTLS 1.2 header by three new content types (`0x1D`, `0x1E`, and `0x1F`). One of those new content types (`0x1E`) represents the MADTLS record layer. The remaining header is extended by a, usually 1-byte long, `segmentation info` field that defines the encrypted payload's segmentation.

This `segmentation info` field starts with two 1-bit flags: The `m`-flag, which is set to enable middlebox authentication (discussed in Section 5.2.5.5), and the `l`-flag, which is set if the layout of the data is explicitly indicated. If the `l`-flag is not set, the

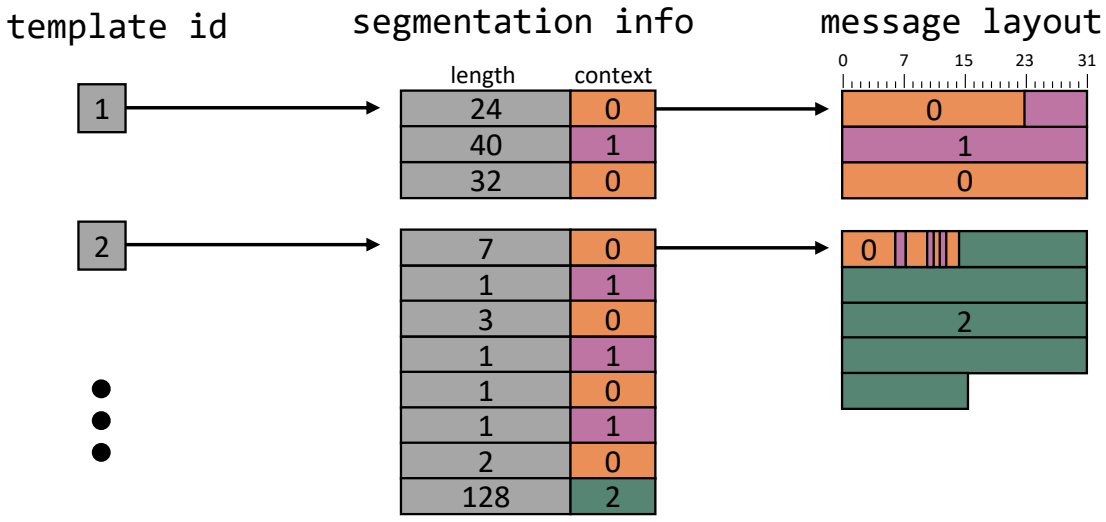


Figure 5.12 The `segmentation info` describes the layout of a packet and can either be explicitly transmitted, or, for repeating patterns, the mapping between a 1-byte `template id` and it can be communicated during the handshake.

remaining 6 bits form the `template id` that maps to one of up to 64 pre-exchanged segmentation infos as shown in Figure 5.12 and explained below.

Conceptually, a plaintext MADTLS message is divided into n segments, addressed as $S[0]$ to $S[n-1]$. Each segment $S[\cdot]$ is assigned a context that defines which middleboxes have read or write access to that segment. The segmentation of a message is then indicated indirectly through the `template id` or by explicitly including the `segmentation info` in the packet header. The `segmentation info` encodes the layout where alternating fields indicate the bitlength of segments and their assigned context. Currently, variable-length fields result in entire segmentation info specifications, but a compact formatting for such cases could also be designed.

As in DTLS, the header precedes the payload that contains the encrypted message and the integrity-protecting authentication tag that is verified by the final receiver of a message. Crucially, MADTLS' authentication tag is updated by middleboxes to ensure data consistency and path integrity. In the following, we discuss the encryption and authentication scheme employed by MADTLS in more detail.

5.2.5.3 Segment Encryption

We start by discussing the encryption scheme used by MADTLS, before diving into the more important (for industrial networks) and challenging aspect of integrity protection. Note that encryption is not mandatory in MADTLS, which may be useful for some (industrial) applications that only care about integrity-protection. When using encryption, the use of a single key obviously does not allow for limited read access. Therefore, each segment $S[\cdot]$ is assigned a context c, which is associated with a unique encryption key k_c^{enc}. Only those middleboxes that should read a specific context are then provided with the corresponding key. These keys are distributed during the MADTLS handshake.

To avoid unnecessary overhead when segments are shorter than a multiple of the block sizes of the cipher (*e.g.*, 16 bytes for AES), MADTLS can take advantage of cipher streams as realized, *e.g.*, by AES in counter mode. This approach ensures that messages do not expand through the encryption of individual segments. Each segment is then separately encrypted with the key corresponding to its context, *i.e.*, $C[i] = enc_{k_c^{enc}}(S[i])$, where $C[i]$ is the encrypted plaintext segment $S[i]$. Middleboxes and the final receiver derive either from the *template id* or from the *segmentation info* field, which segments they have access to, where those are located, and which keys they must use to decrypt which segment.

5.2.5.4 Compact Authentication Scheme

While MADTLS' encryption scheme is straightforward, the analogous approach towards integrity protection is impossible. MADTLS's authentication scheme must differentiate individually between read and write access to all segments. Consequently, we cannot transfer approaches such as mcTLS's three authentication tags [176] to a setting with significantly more fine-granular access control, as this authentication process would introduce three 16-byte tags for each context and does not protect path integrity.

Therefore, we design a custom authentication scheme for MADTLS that ensures compactness, path integrity, and high performance. To achieve these goals, MADTLS takes advantage of authentication tag aggregation [115] to combine multiple tags into a single tag without loss of security for a verifier that is able to verify each of the aggregated tags individually. While tag aggregation has been explored previously to achieve path integrity [73], MADTLS's authentication scheme only needs to transmit a single tag even if messages are modified or divided into multiple contexts. This design naturally favors the verification of data and path integrity by the final receiver, which has access to all contexts to verify all tags. In case the receiver notices that a packet has been manipulated on its path, it can alert the concerned middleboxes or an operator. For now, we focus on this case and describe the design of MADTLS's single authentication tag in the following. We later revisit this limitation in Section 5.2.5.5 and see how MADTLS supports the efficient creation of partial authentication tags that middleboxes with limited data access can still verify.

The gist of our authentication scheme is that each node that is authorized to read or write data manipulates the authentication tag in a deterministic way such that the final tag is correct iff no unauthorized manipulation has taken place during message transmission. The manipulation of the tag by each entity with access rights, even those only allowed to read, ensures that no entity receives manipulated data that is subsequently changed back without this being noticeable by the final receiver. In the following, we describe MADTLS' authentication scheme in detail, starting with the key setup. Afterward, we learn how the sender computes an initial authentication tag and how this tag is subsequently updated by middleboxes according to their access rights.

We have e communication entities and c contexts in a MADTLS session. Each segment $S[\cdot]$ is mapped to a context that uniquely describes a set of access rights for each middlebox. Entities 0 and $e-1$ are the sender and receiver, respectively. All other

		communication entities					
		0	1	2	3	…	$e-2$
contexts	0	$k^{\text{read}}_{0,0}$ ← φ		$k^{\text{read}}_{0,2}$		…	$k^{\text{read}}_{0,e-2}$
	1	$k^{\text{read}}_{1,0}$	$k^{\text{read}}_{1,1}$ ← φ		$k^{\text{read}}_{1,3}$	…	$k^{\text{read}}_{1,e-2}$
	⋮	⋮	⋮	⋮	⋮	⋱	⋮
	$c-1$	$k^{\text{read}}_{c-1,0}$			$k^{\text{read}}_{c-1,3}$	…	$k^{\text{read}}_{c-1,e-2}$

Figure 5.13 MADTLS assigns read keys to communication entities according to their access rights for each context. The same key assignment is conducted for write keys.

entities (1 to $e-2$) are middleboxes with read and/or write access to a subset of all segments. The corresponding symmetric keys for access to the i-th context for the j-th entity are denoted as $k_{i,j}$ as shown in Figure 5.13. For each index, there exist up to two keys, one for read access ($k^{\text{read}}_{i,j}$) and one for write access ($k^{\text{write}}_{i,j}$). As shown in the example of Figure 5.13, Entity 3 has read access to Context 1 such that $k^{\text{read}}_{1,3}$ exists. As Entity 2 has no read access to this Context, $k^{\text{read}}_{1,2}$ does not exist. The sender has read and write access to all contexts of a message. Thus, all keys $k_{\_,0}$ always exist. Meanwhile, no key $k_{\_,e-1}$ exists as the receiver does not need to authenticate the message to another entity. To update authentication tags for the next entity, the k-th communication entity has access to all keys $k_{\_,k}$ that exist. To update (and verify) tags, each entity additionally knows the *previous* existing key for all contexts it has access to. We denote this previous key as $\varphi(k_{\_,k})$. In the example from Figure 5.13, $\varphi(k^{\text{read}}_{0,2}) = k^{\text{read}}_{0,0}$ and $\varphi(k^{\text{read}}_{1,3}) = k^{\text{read}}_{1,1}$.

In the following, we explain how the sender uses the keys $k_{\_,0}$ to compute the initial authentication tag. Each entity subsequently updates the aggregated authentication tags by removing the old partial segment tag (*i.e.*, the authentication tag computed only over a segment with the corresponding read or write key) and adding a new tag for each accessed context. As each update requires access to a unique set of keys that only that specific middlebox knows, it cannot be impersonated by another entity. An in-depth discussion of the soundness and security of MADTLS' authentication scheme is provided in Section 5.2.9.

The initial tag is computed as follows, where $\delta(\cdot)$ maps the segment index i to the corresponding context:

$$t = \bigoplus_{0 \le i < n} \left(\sigma_{k^{\text{read}}_{\delta(i),0}}(C[i]) \oplus \sigma_{k^{\text{write}}_{\delta(i),0}}(C[i]) \right)$$

Each segment is authenticated (σ represents a classical message authentication algorithm such as HMAC-SHA256) twice, with the corresponding reading and writing keys, respectively. All computed tags (*i.e.*, reading and writing tags for all segments) are then XOR-ed together, which does not reduce their security [31]. A verifier now needs all keys that were used to compute the individual tags to verify the aggregated tag. This aggregated tag is then appended to the message and transmitted to the first middlebox with limited read or write access to the message.

All middleboxes alter this tag according to their access rights in a deterministic way. A middlebox j that has read access to the segments $\mathbb{S}_j^{\text{read}}$ updates the tag of the message as follows:

$$t \overset{\oplus}{=} \bigoplus_{i \in \mathbb{S}_j^{\text{read}}} \left(\sigma_{\varphi(k^{\text{read}}_{\delta(i),j})}(C[\![i]\!]) \oplus \sigma_{k^{\text{read}}_{\delta(i),j}}(C[\![i]\!]) \right)$$

For each segment i in $\mathbb{S}_j^{\text{read}}$, the first part of the equation removes tags from the last entity that had read access to that message segment. This removal works exactly when the segment $C[\![i]\!]$ has not been changed between the two readers, as only then do the old partial segment tag and the newly computed partial segment tag cancel out. The second part of the equation then computes and integrates a new segment tag with the new key $k^{\text{read}}_{\delta(i),j}$ not known to the previous middlebox.

Besides read access, some middleboxes may also have write access to certain segments $\mathbb{S}_j^{\text{write}}$. Here, we assume that write access implies read access. Formally, this means that $\mathbb{S}_j^{\text{read}} \cap \mathbb{S}_j^{\text{write}} = \varnothing$, *i.e.*, write access to a context cannot be combined with explicit read access. Here, the procedure to amend the authentication tag is similar. First, the old *reading and writing* tags are removed before they are replaced by the new tag computed over the changed segment data $C'[\![\cdot]\!]$. Formally, this procedure looks as follows:

$$t \overset{\oplus}{=} \bigoplus_{i \in \mathbb{S}_j^{\text{write}}} \Big(\sigma_{\varphi(k^{\text{read}}_{\delta(i),j})}(C[\![i]\!]) \oplus \sigma_{k^{\text{read}}_{\delta(i),j}}(C'[\![i]\!]) \oplus \sigma_{\varphi(k^{\text{write}}_{\delta(i),j})}(C[\![i]\!]) \oplus \sigma_{k^{\text{write}}_{\delta(i),j}}(C'[\![i]\!]) \Big)$$

Finally, the receiver receives the message as well as the transmitted and updated authentication tag t. Based on the received data, it can then compute t^* as follows:

$$t^* = \bigoplus_{0 \le i < n} \left(\sigma_{\varphi(k^{\text{read}}_{\delta(i),e-1})}(C[\![i]\!]) \oplus \sigma_{\varphi(k^{\text{write}}_{\delta(i),e-1})}(C[\![i]\!]) \right)$$

If no unauthorized manipulation of transmitted data took place, t and t^* are identical, and thus the integrity of the message is verified successfully. Otherwise, at least one communication entity has been served with unauthentic data.

5.2.5.5 Self-Verifying Middlebox

By default, MADTLS operates in the most resource-conscious mode where only the final receiver verifies a message. Notably, this verification not only covers the authenticity of the message as received at the final destination but also ensures the authenticity of the message as received by each on-path middlebox (with read or write permission). However, further efforts are required to ensure on-path authenticity if the processing of messages by middleboxes causes *side effects*, *i.e.*, has influences beyond the current message, *e.g.*, the injection of a control command.

Here, a middlebox cannot always exclusively rely on the final receiver to authenticate a message. In some cases, *optimistic processing* [246, 269] (*i.e.*, the idea of processing

a likely genuine message with the knowledge that maliciousness is guaranteed to be detected within a short time span) still allows offloading authentication to the final receiver for reliable connections (*e.g.*, TCP). But at the very least, middleboxes with side effects that communicate over lossy channels must authenticate the data they process themselves. Otherwise, an attacker could manipulate a message before a middlebox processes it and prevent it from being received at its final destination, *e.g.*, by jamming a wireless channel.

In cases where *immediate* verification of authenticity is required (*e.g.*, for irreversible critical decisions), MADTLS enables such middleboxes to *self-verify* authenticity by specifically adding an additional 16-byte authentication tag t_j for the middlebox j. This tag t_j is computed over the subset of partial tags σ relevant to the middlebox's access rights, such that no additional cryptographic processing is necessary. Formally, it is computed as follows:

$$t_j = \bigoplus_{i \in \mathbb{S}_j^{\text{read}} \cup \mathbb{S}_j^{\text{write}}} \left(\sigma_{k_{\delta(i),0}^{\text{read}}}(C[i]) \right) \oplus \bigoplus_{i \in \mathbb{S}_j^{\text{write}}} \left(\sigma_{k_{\delta(i),0}^{\text{write}}}(C[i]) \right)$$

Like the main authentication tag t, these tags t_j are modified by preceeding middleboxes. These modifications are, however, restricted to the subset of contexts both middleboxes have access to, which is communicated to the middleboxes during the handshake. For write access, a middlebox k would modify the tag t_j as follows:

$$t_j \overset{\oplus}{=} \bigoplus_{i \in \mathbb{S}_j^{\text{write}} \cap \mathbb{S}_k^{\text{write}}} \Big(\sigma_{\varphi(k_{\delta(i),k}^{\text{read}})}(C[i]) \oplus \sigma_{k_{\delta(i),k}^{\text{read}}}(C'[i]) \oplus \sigma_{\varphi(k_{\delta(i),k}^{\text{write}})}(C[i]) \oplus \sigma_{k_{\delta(i),k}^{\text{write}}}(C'[i]) \Big)$$

Analogously, the intermediary middlebox k only updates the read tags if it and middlebox j have access to a context:

$$t_j \overset{\oplus}{=} \bigoplus_{i \in \mathbb{S}_j^{\text{read}} \cap (\mathbb{S}_k^{\text{read}} \cup \mathbb{S}_k^{\text{write}})} \left(\sigma_{\varphi(k_{\delta(i),k}^{\text{read}})}(C[i]) \oplus \sigma_{k_{\delta(i),k}^{\text{read}}}(C'[i]) \right)$$

Finally, upon reception of the message by middlebox j, this middlebox can verify the authenticity of the data it has access to by recomputing t_j and comparing it to the transmitted tag. When forwarding the message to the final receiver, middlebox j removes the tag t_j as it is no longer needed. After learning about self-verifying middleboxes, we can now look at how MADTLS enables limited message injection as required by some industrial use cases.

5.2.5.6 Limited Message Injection Capabilities

So far, MADTLS allows authorized middleboxes to read and write to well-defined segments of transmitted messages. Still, certain industrial use cases for middleboxes additionally require capabilities to actively inject new messages, *e.g.*, to enable low latency control of robot arms (cf. Section 5.2.2.1). However, giving such middleboxes

the ability to inject *arbitrary* messages breaks the least-privilege principle, as a middlebox could take control beyond what is necessary for its dedicated task. Consequently, MADTLS needs to enforce *limited injection capabilities* where middleboxes can only inject those messages relevant to its intended functionality.

MADTLS realizes these injection capabilities through the use of pre-defined *message templates*. In essence, a template is a message with dedicated placeholders that is authenticated by the endpoint and sent to the middlebox in advance. The middlebox then has restricted write access to the placeholder segments of this message before it is forwarded (*e.g.*, to only be able to transmit specific control commands).

However, DTLS uses nonces that are explicitly transmitted to prevent replay attacks. Either way, for messages generated via message templates, we must ensure that *(1)* the used nonce is unique and *(2)* the receiver knows and accepts the nonce. In DTLS, nonces are a combination of a sequence number and an epoch. To not interfere with this procedure, MADTLS defines a new content type (`0x1F`) to mark middlebox-injected messages. These messages use the same encryption and authentication keys as normal messages, but start at any unique epoch in the future. This epoch should be chosen according to how many injected messages are expected in relation to normal messages and how many message-injecting middleboxes exist in a given communication session.

While a middlebox with the mere task of sending emergency stops in critical situations only needs the ability to send a few messages, a middlebox responsible for low-latency control adjustments continuously injects a significant number of messages. As the sequence numbers of injected and regularly transmitted messages (from the original sender) are decoupled, the impersonated endpoint can asynchronously provide multiple authentication tags in advance for increasing sequence numbers of the same message template without having to retransmit the template itself. Hence, MADTLS achieves the same fine-grained access control over injected messages as for read and write access to existing messages.

5.2.6 The MADTLS Handshake Protocol

MADTLS requires several keys to enable fine-grained access control for middleboxes. These keys can be preconfigured or distributed ad hoc by a trusted party. In many cases, it is, however, beneficial if the involved parties can agree upon these keys by themselves. Therefore, we adapt the DTLS 1.2 handshake to additionally exchange the keys required for MADTLS to the endpoints as well as all involved middleboxes. For simplicity, we assume that all communication entities have pre-established shared secrets, which is often reasonable as all entities are known in advance and managed by a single operator in industrial networks. To enable certificate-based authentication of all entities, we can employ the `ServerKeyExchange` message proposed in step 3 of the mcTLS handshake [176], which we, however, do not require due to our pre-shared keys. Figure 3.21 highlights the necessary additions to the DTLS 1.2 handshake in blue. In the following, we discuss these changes step by step.

`ClientHello`. The `ClientHello` announces the desire to establish a new communication session and proposes a set of cipher suites. The `ClientHello` contains a

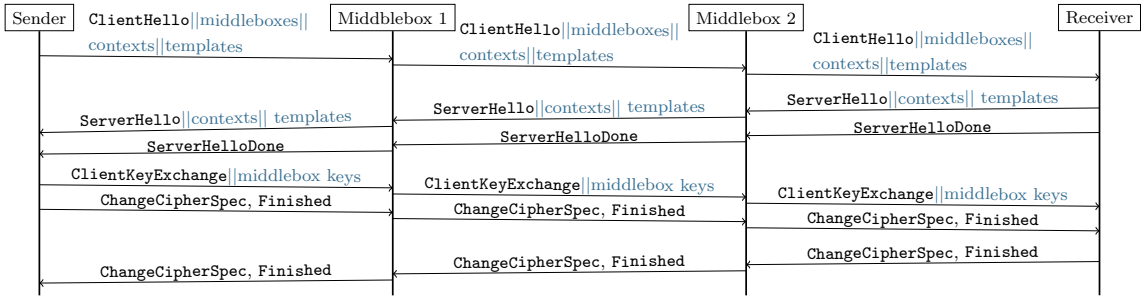


Figure 5.14 For MADTLS, we extend the DTLS 1.2 handshake and add additional data to select messages (highlighted in blue).

pre-exchanged cookie to thwart DoS attacks as per the DTLS 1.2 standard, which we do not change. One important change is that the `ClientHello` message passes through all middleboxes, which are thus informed about the new session and may remove cipher suites from the announced list. We extend it by the information concerning middleboxes and their access rights. As additional information, MADTLS first appends a list of middleboxes identified by their IP address. Then, the `contexts` field defines all contexts needed by that session, *i.e.*, combinations of read and write access for the different middleboxes. Finally, the client appends a list of possible message templates. First, the number of templates is announced, followed by a null-byte terminated enumeration of `segmentation info` fields, as introduced in Figure 5.12. If the receiver accepts the request and supports a requested cipher suite, it responds with a `ServerHello` message.

`ServerHello`. The receiver responds with a `ServerHello` handshake message, agrees on a selected cipher suite, and replays the MADTLS-specific *contexts* and *template* fields. These fields must be replayed as the server may add more contexts or templates to them.

`ServerHelloDone`. The `ServerHelloDone` does not require any modification as it only signals the end of the `ServerHello`.

`ClientKeyExchange`. The client then shares the final information the server needs to derive the symmetric keys used in this session and shares all middlebox keys k^{read} and k^{write} with the respective involved middleboxes through a `ClientKeyExchange` message. For this key distribution, the sender first computes a key to encrypt and authenticate data to the respective middleboxes. Based on $\text{secret}_{s,m}$ shared between the sender s and a middlebox m, these key distribution keys k^{kd} are derived as:

$$k_m^{kd} = \text{KDF}(\text{secret}_{s,m}, \text{nonce}),$$

where the nonce stems from the DTLS 1.2 handshake. Then, the context encryption keys k^{enc} are derived independently of any secrets shared with a middlebox, as:

$$\text{KDF}(\text{secret}_{s,r}, \text{nonce}, \text{context}, \text{`encrypt'})$$

from the secret shared between sender s and receiver r, the nonce, an identifier of the respective context and a unique string. Finally, the context authentication keys k^{read} and k^{write} are derived as:

$$\text{KDF}(\text{secret}_{s,r}, \text{nonce}, \text{middlebox}, \text{context}, \{\text{`read'}, \text{`write'}\})$$

by including an identifier of the targeted middlebox and different strings for read or write keys. To ensure that a middlebox only gains access to its keys (and the respective preceding keys as given by $\varphi(\cdot)$), the client encrypts each middlebox's context keys with the respective key distribution key k_m^{kd} before appending them to the `ClientKeyExchange` message. For example, consider that middlebox 2 has read access to context 1 and write access to context 3. Then, the sender would append $\text{enc}_{k_2^{kd}}(k_{1,2}^{\text{read}} \parallel k_{3,2}^{\text{read}} \parallel k_{3,2}^{\text{write}} \parallel \varphi(k_{1,2}^{\text{read}}) \parallel \varphi(k_{3,2}^{\text{read}}) \parallel \varphi(k_{3,2}^{\text{write}}))$ to the `ClientKeyExchange` message. The middleboxes can then decrypt these respective keys as the `ClientKeyExchange` passes. The subsequently transmitted `ChangeCipherSpec` and `Finished` messages are not changed for MADTLS.

Final Server Messages. In DTLS 1.2, the server sends a final copy of `ChangeCipherSpec` and `Finished` messages to conclude the handshake. This procedure is not changed by MADTLS as the receiver can compute and verify all key distribution keys that the sender transmitted in the `ClientKeyExchange` message. Overall, a MADTLS session can be efficiently established, and we now look at its performance in real-world scenarios.

5.2.7 Performance Evaluation

MADTLS fulfills the outlined functional requirements (*cf.* Section 5.2.3.2) expected of a middlebox-aware security protocol for industrial networks. To evaluate if its performance is suitable even for resource-constrained IIoT devices, we implemented a prototype for Contiki-NG 4.8, a popular operating system for IoT devices, by extending the *tinydtls* library.

5.2.7.1 MADTLS vs. the Current State-of-the-Art

To begin our evaluation, we compare the processing latency of MADTLS to related approaches. These approaches consist of the tinydtls DTLS 1.2 implementation (which offers no middlebox awareness) and a custom mcTLS implementation[4] adapted to DTLS to avoid TCP overhead. For all protocols, we send payloads of lengths increasing from 1 to 256 bytes. MADTLS uses a single write context to emulate the same functionality as mcTLS. Our measurements run on a Zolertia RE-Mote (Cortex M3 @ 32 MHz, 32-bit CPU), a common device to represent the constraints of industrial hardware [203, 246, 250].

Figure 5.15 plots the processing time against the length of the transmitted payload for the different protocols. All protocols require more processing as the payload increases. However, these increases are discrete, marginally rising by about 0.1 ms whenever the payload fills up a new AES block (all 16 bytes), and more substantially jumping by about 0.4 ms when sha256 blocks of the HMAC computation are filled up (all 64 bytes). Thus, MADTLS is efficient even for large messages and even

[4] We could not source any implementations of the approaches and therefore implement the closest competitor of MADTLS ourselves.

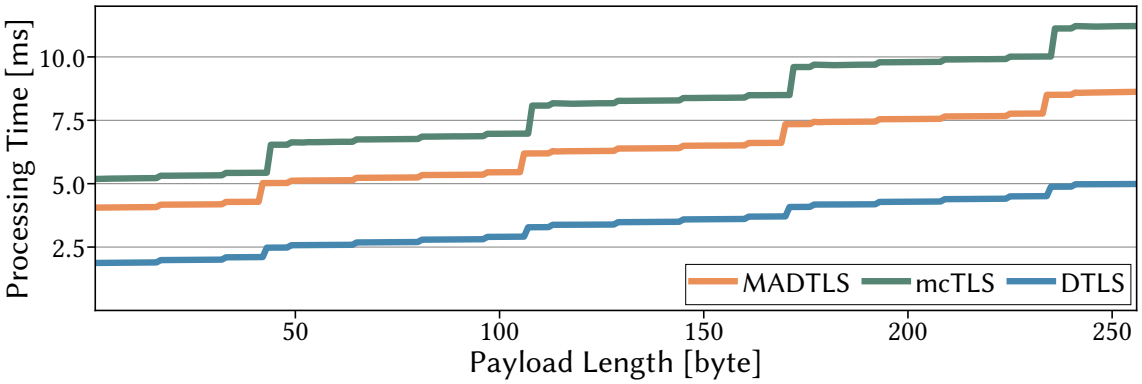


Figure 5.15 MADTLS performs better than mcTLS in comparable scenarios, where middlebox write access is given for selected messages. DTLS provides no such guarantees.

achieves a noticeable performance gain against mcTLS, which neither offers the same fine-grained data access as MADTLS nor ensures path integrity.

Beyond latency, reducing bandwidth usage is also imperative for MADTLS. Here, the DTLS 1.2 record protocol carries a header overhead (including 16 bytes tags) of 30 bytes while mcTLS's overhead is 63 bytes. Meanwhile, MADTLS only adds 1 byte to DTLS 1.2 for the `template id` that defines the message structure. Even for multiple contexts, MADTLS does not require more space.

MADTLS is thus attractive performance-wise for industrial networks, even if no fine-grained data access is needed. In the case of a single context, it saves over 22 % of processing latency (more for larger messages) and 32 bytes of header over mcTLS. Most importantly, MADTLS offers this performance while offering true least-privilege data access to middleboxes, ensuring path integrity, and allowing least privilege traffic injection, three crucial features not offered by mcTLS, or any other approach from related work.

Cipher Suite for Faster Processing. MADTLS performs adequately for many industrial applications. However, some applications require even faster processing in the sub-millisecond range. To assess MADTLS under these harsh requirements, we design a cipher suite to minimize processing latency based on recent work on preprocessed encryption and integrity protection. More concretely, we create a cipher suite based on antedated encryption [101] and BP-MAC [250]. Antedated encryption uses AES in counter mode and splits it into a precomputation phase for the computationally-intensive keystream generation and a fast online phase that only XORs a message with the cached keystream [101]. BP-MAC, in turn, achieves fast integrity protection for short messages by combining precomputed authentication tags for individual bits via a Carter-Wegman construction [250]. Using this cipher suite for MADTLS on our resource-constrained RE-Mote board, message encryption and authentication time reduces to 624 µs (compared to 3.975 ms) for a 5-byte message with a single write context spanning the entire message. Thus, with suitable cryptographic algorithms, MADTLS is apt for low-latency scenarios on constrained hardware.

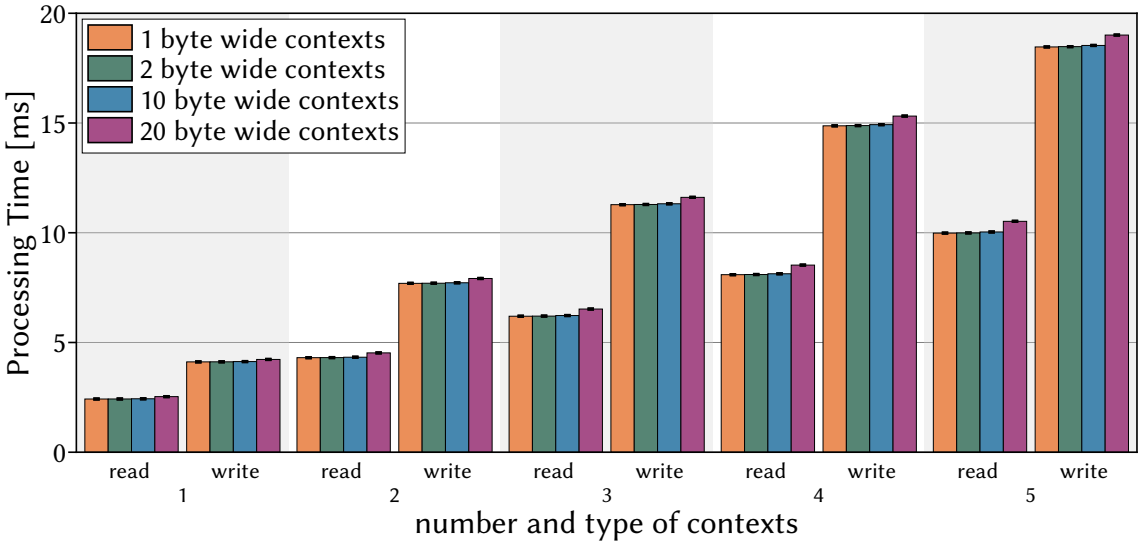


Figure 5.16 MADTLS's performance scales linearly in the number of contexts, while their sizes only have a marginal impact.

5.2.7.2 Impact of the Size and Number of Contexts

We must also understand the impact of MADTLS' fine-grained access control on its performance. Therefore, we evaluate the impact of the number and size of contexts on the processing time. We continue using the Zolertia RE-Mote as evaluation platform and measure the encryption and authentication time of a 100-byte long packet. We add 1 to 5 contexts of sizes varying between 1 and 20 bytes to each message. We repeat our measurements 20 times and show the results, including 99%-confidence intervals, in Figure 5.16.

We observe a linear growth of processing times with the number of contexts, and that write contexts require more processing than read contexts. This scaling is expected as it is proportional to the number of calls to HMAC-SHA256. While processing one 20-byte read context takes 2.42 ms, the same operation over a write context lasts 4.22 ms. This behavior can also be explained by the number of HMAC-SHA256 calls, as write contexts require two calls per context compared to the one required for read contexts. Meanwhile, the size of the contexts only has a negligible impact on the processing time, as all contexts individually fit into a single SHA256 block.

Overall, MADTLS performs adequately even for scenarios requiring many contexts for different middleboxes with diverse functionalities. As contexts are shared if the same data is accessed by multiple middleboxes, even complex chains of middleboxes can operate in a least-privilege mode with a low number of contexts. Thus, even resource-constrained devices, commonly seen as endpoints in industrial networks, can employ MADTLS.

5.2.7.3 MADTLS Across Different Hardware Classes

Middleboxes are typically more powerful as they have to process many messages from different sources. Therefore, we use the Raspberry Pi Zero (which uses the same

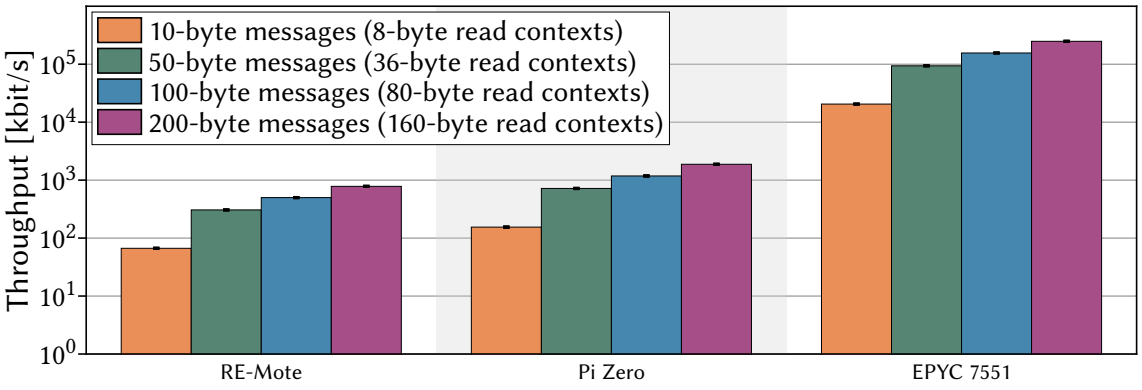


Figure 5.17 Throughput is limited on constrained devices, but middleboxes can take advantage of more powerful hardware.

ARM 11 chip that the Netronome NFP-4000 Flow Processor uses for general-purpose processing) to evaluate the performance of SmartNIC-based middleboxes without further optimizations. Moreover, we evaluate MADTLS on an AMD EPYC 7551 server CPU. Thus, we can learn how MADTLS performs over a wide variety of CPU classes and architectures.

For our evaluation, we process 10, 50, 100, and 200 byte messages with the middlebox accessing 80 % of a message's data via a single read context. The processing entails the decryption of the accessed data and updating the authentication tag. We repeat all measurements 20 times and report on the throughput with 99%-confidence intervals in Figure 5.17. Across all processors, the throughput grows significantly with longer messages as the fixed per-message overhead mainly impacts small messages.

Our evaluation shows that if devices like the Zolertia RE-Mote were used as a middlebox, they would still achieve a throughput of over 66 kbit/s. The Raspberry Pi Zero more than doubles this throughput, with a throughput between 154 and 1877 kbit/s, depending on the message sizes. Expectedly, SmartNICs (as well as programmable switches) can, however, not rely on a relatively slow general-purpose processing core to achieve gigabit throughput. Still, their performance can be significantly optimized with device-specific implementations of cryptographic primitives [124, 264]. On the other hand, our AMD EPYC 7551 processor achieves a throughput between 20 and 249 Mbits/s on a single one of its 32 cores.

5.2.7.4 MADTLS in the Real World

To verify MADTLS's utility in real-world use cases, we conduct two case studies. First, we implement a middlebox that translates between local coordinates of robot arms and global coordinates as proposed by Kunze *et al.* [129]. Here, MADTLS allows us to ensure that the middlebox only accesses the first six byte of a packet's payload where the x, y, and z coordinates are encoded in 16 bits each. The remaining payload, containing supplementary sensor readings (*e.g.*, timestamp, grip pressure, gripper rotation) in an additional 14 bytes, cannot be written to or read by the middlebox. This data layout, *e.g.*, corresponds to one observed in Modbus communications.

Secondly, we use MADTLS to realize an IDS based on Snort rules. To this end, we use the 14 Quickdraw Snort rules [5] for industrial Modbus communication. Using MADTLS, we constrain the IDS's access to selectively reading 3, 5, or 6 bytes per Modbus frame, depending on the communication flow. Without restricting the IDS's capabilities, MADTLS thus blinds over 60 % of all bytes in the corresponding Modbus test traces [5].

5.2.8 Limitations of MADTLS

MADTLS addresses many shortcomings of related work on secure middlebox-aware (industrial) communication. These achievements come, however, with a few drawbacks that must be considered before deploying MADTLS. First, MADTLS' handshake is significantly more complex than the standard DTLS 1.2 handshake. While it does not require additional round trips, more data must be exchanged between the entities. Per se, this is not a problem in industrial networks with relatively static connections since handshakes can be performed in advance during non-critical periods. However, the added complexity makes weaknesses in design and implementation more likely and may be restrictive in some scenarios.

Secondly, MADTLS needs more extensive key management and storage than simple end-to-end communication. While this drawback is inevitable when multiple entities with different access rights on a single communication channel are involved, it may impact embedded devices with limited storage capabilities. Also, the corresponding key exchange protocol, as exemplified in Section 5.2.6, becomes more error-prone in terms of design and implementation.

Thirdly, MADTLS offloads integrity verification to the final receiver of a message by default. However, it must be carefully considered whether middleboxes must additionally verify the integrity of processed data themselves. Still, MADTLS offers efficiently computed additional middlebox-specific authentication tags that require no additional cryptographic processing to append to a message.

Fourthly, while MADTLS is designed with performance in mind and even outperforms its closest competitor (mcTLS [176]), the added processing overhead is not negligible for resource-constrained devices in industrial scenarios. Fortunately, MADTLS's processing adds minimal jitter, such that a deterministic overhead can be considered when designing control algorithms where required [209].

Fifthly, MADTLS adopts DTLS 1.2 with our building blocks to bring middlebox-aware security to industrial communication. In reality, the industrial landscape and beyond also uses other security protocols (*e.g.*, TLS) that can and should not always be replaced with DTLS. While we see nothing preventing the adoption of other protocols with MADTLS's features, a concrete design must still be proposed to bring our advances to a wider variety of applications.

Sixthly, MADTLS is most efficient for predictable message structures as they are often found in the IIoT. Employing MADTLS on the traditional Internet would expose it to more dynamic content (*e.g.*, websites). While the explicit transmission

of segmentation info enables such scenarios, the added bandwidth overhead must be considered. Moreover, privacy implications of metadata (*e.g.*, for templates) must be carefully considered in other scenarios, as devices no longer belong to a single operator in that case.

MADTLS thus mostly exhibits limitations outside the industrial domain. However, domain-specific adaptations can alleviate some of these constraints while benefiting from MADTLS's main contributions: Allowing least-privilege access control on a communication channel for middlebox processing.

5.2.9 Security and Soundness Proofs of Madtls

In the following, we formally prove the security of the MADTLS protocol against inside and outside attackers. Before that, we also prove the soundness of MADTLS.

5.2.9.1 Soundness of Madtls

We first prove the soundness of MADTLS's authentication scheme, *i.e.*, that without malicious manipulation all valid tags are accepted. First, we look at the case when no middlebox is included.

$$\begin{aligned} t^* &= \bigoplus_{0 \le i < n} \left(\sigma_{\varphi(k^{\text{read}}_{\delta(i),j-1})}(X[i]) \oplus \sigma_{\varphi(k^{\text{write}}_{\delta(i),j-1})}(X[i]) \right) \\ &= \bigoplus_{0 \le i < n} \left(\sigma_{k^{\text{read}}_{\delta(i),0}}(X[i]) \oplus \sigma_{k^{\text{write}}_{\delta(i),0}}(X[i]) \right) \\ &\overset{!}{=} t \end{aligned}$$

Then, we prove by induction that the inclusion of a reading middlebox k as last hop before the intended receiver does transform a valid tag t' into a valid final tag t^*.

$$\begin{aligned} t' = & \bigoplus_{0 \le i < n} \left(\sigma_{\varphi(k^{\text{read}}_{\delta(i),k})}(X[i]) \oplus \sigma_{\varphi(k^{\text{write}}_{\delta(i),k})}(X[i]) \right) \\ \overset{\text{read}}{=} & \bigoplus_{0 \le i < n} \left(\sigma_{\varphi(k^{\text{read}}_{\delta(i),k})}(X[i]) \oplus \sigma_{\varphi(k^{\text{write}}_{\delta(i),k})}(X[i]) \right) \oplus \\ & \bigoplus_{i \in \mathbb{S}^{\text{read}}_k} \left(\sigma_{\varphi(k^{\text{read}}_{\delta(i),k})}(X[i]) \oplus \sigma_{k^{\text{read}}_{\delta(i),k}}(X[i]) \right) \\ = & \bigoplus_{i \notin \mathbb{S}^{\text{read}}_k} \left(\sigma_{\varphi(k^{\text{read}}_{\delta(i),k})}(X[i]) \oplus \sigma_{\varphi(k^{\text{write}}_{\delta(i),k})}(X[i]) \right) \oplus \\ & \bigoplus_{i \in \mathbb{S}^{\text{read}}_k} \left(\sigma_{k^{\text{read}}_{\delta(i),k}}(X[i]) \oplus \sigma_{\varphi(k^{\text{write}}_{\delta(i),k})}(X[i]) \right) \\ = & \bigoplus_{0 \le i < n} \left(\sigma_{\varphi(k^{\text{read}}_{\delta(i),j-1})}(X[i]) \oplus \sigma_{\varphi(k^{\text{write}}_{\delta(i),j-1})}(X[i]) \right) \\ \overset{!}{=} & t^* \end{aligned}$$

Finally, we do the same inductive proof for a writing middlebox.

$$
\begin{aligned}
t' = & \bigoplus_{0\le i<n} \left(\sigma_{\varphi(k^{\text{read}}_{\delta(i),k})}(X[i]) \oplus \sigma_{\varphi(k^{\text{write}}_{\delta(i),k})}(X[i])\right) \\
\overset{\text{write}}{=} & \bigoplus_{0\le i<n} \left(\sigma_{\varphi(k^{\text{read}}_{\delta(i),k})}(X[i]) \oplus \sigma_{\varphi(k^{\text{write}}_{\delta(i),k})}(X[i])\right) \oplus \\
& \bigoplus_{i\in\mathbb{S}^{\text{write}}_k} \left(\sigma_{\varphi(k^{\text{read}}_{\delta(i),k})}(X[i]) \oplus \sigma_{k^{\text{read}}_{\delta(i),k}}(X'[i]) \oplus \right. \\
& \qquad \left. \sigma_{\varphi(k^{\text{write}}_{\delta(i),k})}(X[i]) \oplus \sigma_{k^{\text{write}}_{\delta(i),k}}(X'[i])\right) \\
= & \bigoplus_{i\notin\mathbb{S}^{\text{write}}_k} \left(\sigma_{\varphi(k^{\text{read}}_{\delta(i),k})}(X'[i]) \oplus \sigma_{\varphi(k^{\text{write}}_{\delta(i),k})}(X'[i])\right) \oplus \\
& \bigoplus_{i\in\mathbb{S}^{\text{write}}_k} \left(\sigma_{k^{\text{read}}_{\delta(i),k}}(X'[i]) \oplus \sigma_{k^{\text{write}}_{\delta(i),k}}(X'[i])\right) \\
= & \bigoplus_{0\le i<n} \left(\sigma_{\varphi(k^{\text{read}}_{\delta(i),j-1})}(X'[i]) \oplus \sigma_{\varphi(k^{\text{write}}_{\delta(i),j-1})}(X'[i])\right) \\
\overset{!}{=} & \, t^*
\end{aligned}
$$

5.2.9.2 Security of MADTLS

Besides the soundness of MADTLS's authentication scheme, its security is also crucial. To prove its security, we first look at the security definition of traditional MAC schemes to show that MADTLS cannot be attacked by outsiders. Afterward, we also prove that an insider, *i.e.*, a middlebox authorized to read or write some part of a packet, cannot manipulate a packet beyond what it is authorized to do.

5.2.9.2.1 MADTLS is Secure Against Outsiders

Our proof is built upon the fact that $\mathrm{Adv}_{\mathcal{B}}[\Sigma^{\oplus}] \le \mathrm{Adv}_{\mathcal{A}}[\Sigma]$, where a tag t by $\Sigma^{\oplus}$ is computed as $t = \Sigma(X[1]) \oplus \cdots \oplus \Sigma(X[n])$, with $m = X[1]||\ldots||X[n]$. We first adapt Attack Game 1 (*cf.* Section 2.3.1) to prove the security of a MAC scheme that allows (partial) manipulations by authorized middleboxes.

Attack Game 2

- $\mathcal{C}$ randomly picks a key k from $\mathcal{K}$.
- The adversary queries an oracle with a segment $X_i[\cdot]$ for a valid partial tag $t_i[\cdot]$, *i.e.*, $t_i[\cdot]$ such that $\mathrm{Vrfy}^{\Sigma}_k(X_i[\cdot], t_i[\cdot])$ returns `accept`. Denote $\mathbb{M}$ the set of all m_i that can be composed of partial messages queried by $\mathcal{A}$.
- Eventually, $\mathcal{A}$ outputs a candidate forgery $(m, t) \in \mathcal{M} \times \mathcal{T}$, with $m \notin \mathbb{M}$.

In Attack Game 2, the adversary is strictly more powerful than in Attack Game 1, as they can query partial authentication tags, but still (over multiple queries) learn the

tag over every specific message. Thus, MADTLS is secure against outsider attacks if an adversary $\mathcal{A}$ cannot win Attack Game 2 with a non-negligible probability.

Still, the advantage of $\mathcal{A}$ to win Attack Game 2 for $\Sigma^{\oplus}$ remains negligible: For every $m \notin \mathbb{M}$, there exists at least one m_i for which t_i was not queried by $\mathcal{A}$. Thus, $\mathrm{Adv}_{\mathcal{B}}[\Sigma^{\oplus}]$ can be at most $\mathrm{Adv}_{\mathcal{A}}[\Sigma]$, as otherwise $\mathcal{A}$ could learn t_i for $X_i[\cdot]$ which were not queried, *i.e.*, $\mathrm{Adv}_{\mathcal{B}}[\Sigma^{\oplus}] \leq \mathrm{Adv}_{\mathcal{A}}[\Sigma]$. Consequently, MADTLS's authentication scheme is secure as long as the underlying MAC scheme is secure and the tags t_i computed for different message segments $X_i[\cdot]$ are independent.

5.2.9.2.2 MADTLS is Secure Against Insiders

We established thus far that only authorized writers can alter (segments of) messages, such that they are successfully verified by the receiver. However, we have not considered the possibility of ephemeral changes by malicious actors, *i.e.*, the temporary altering of a message for only one or a few middleboxes' processing, before the message is altered back to a message accepted by the final receiver. To validate that such an attack is not possible, we have to take a deeper look at how a message's authentication tag changes over time.

When middlebox j accesses a segment, it alters the authentication tag in a deterministic way: $t \oplus \sigma_{\varphi(k_{\_,j})}(X[\cdot]) \oplus \sigma_{k_{\_,j}}(X[\cdot])$. The partial tags from the last accessing entity are first removed from the aggregated tag, before new partial tags are added by the current entity. This process might be done once or twice, depending on whether the accessing middlebox is only reading or also writing.

Consequently, if an attacker maliciously changes the message segment $X'[\cdot]$, they must be able to compute $\sigma_{\varphi(k_{\_,j})}(X'[\cdot])$ and $\sigma_{k_{\_,j}}(X'[\cdot])$ in order to remove the consequences of their attack from the authentication tag. It is, however, precisely the middlebox j doing these computations that has access to the required keys (besides the endpoints of the communication channel). The next middleboxes accessing that specific segment may compute $\sigma_{k_{\_,j}}(X'[\cdot])$ for the attacker if the message has not been changed back yet. However, then $\sigma_{\varphi^{-1}(k_{\_,j})}(X'[\cdot])$ needs to be intercepted by the attacker. In the end, the attacker needs to compute two partial authentication tags to ephemerally change a message in an unauthorized way. Thus, the collusion of at least two middleboxes having read access to a specific message segment is required for an attack that introduces ephemeral changes. Hence, no single actor, external or internal, can attack MADTLS's authentication scheme.

5.2.10 Summary

This thesis introduces MADTLS, a middlebox-aware enhancement of the DTLS protocol tailored explicitly to the IIoT. Hereby, MADTLS addresses multiple major limitations of the current state-of-the-art on middlebox-aware security that focuses heavily on the traditional Internet. Most importantly, MADTLS allows fine-grained access control for middleboxes on a bit-level and enables middleboxes to inject a

restricted set of messages where this is desired (*e.g.*, for emergency shutdowns) while still operating more efficiently than its closest competitor mcTLS [176].

Concretely, MADTLS segments messages and assigns contexts (*i.e.*, read and write access rights) according to the middleboxes' needs. Each segment is encrypted and authenticated separately without expanding the packet. Middleboxes are permitted to read or write to specific segments and either verify integrity directly or defer this step to the final receiver for efficiency reasons. MADTLS processes packets in only a few milliseconds on heavily constrained hardware, while performance scales linearly with the number of contexts and message lengths. Meanwhile, MADTLS achieves up to 249 Mbit/s throughput on a single AMD EPYC 7551 core, enabling middleboxes to process many different communication streams.

Thus, MADTLS introduces middlebox-aware security to the IIoT to solve the dilemma of middleboxes becoming increasingly popular to improve efficiency and thereby preventing secure end-to-end communication.

5.3 Conclusion

To protect the authenticity of messages in a communication system with more than two participants, typically, asymmetric cryptographic primitives, such as digital signatures, are employed. These primitives bring significant processing and bandwidth demands, leaving resource-constrained IIoT devices often without proper means for adequate protection. With CAIBA and MADTLS, this thesis introduces two protocols to adapt symmetric MAC schemes to such group communication systems. In contrast to prior work, CAIBA and MADTLS neither introduce verification delay nor occupy additional bandwidth compared to ordinary group key-based authentication.

The contributions from this chapter thus demonstrate how long-standing limitations can be overcome by focusing on distinct scenarios. While CAIBA is not as generally applicable as competing schemes such as the TESLA protocol [188, 189], its applicability to the CAN bus alone could protect billions of vehicles against masquerading attacks as used by, *e.g.*, the CAN Invader [3]. Similarly, MADTLS focuses on specifically the challenge of integrating middleboxes into end-to-end protected channels in limited domains (*cf.* Section 5.2.2.2). Thus, MADTLS lays the foundation to bring end-to-end security to ICSs without forgoing the immense improvements introduced by middleboxes in this domain, and thus offers protection against the increasing number of targeted cyberattacks against these networks.

6

“ Don’t adventures ever have an end? I suppose not. Someone else always has to carry on the story. ”

Bilbo Baggins, *The Fellowship of the Ring*, 1954

Conclusion

The global increase of cyberattacks against Industrial Control Systems (ICSs) in critical infrastructures demands an urgent response. In the past, these networks were physically isolated and correspondingly harder to attack. However, increased remote connectivity, advanced attacks, and heightened interest by state-level adversaries increasingly blur this physical line of defense.

Consequently, the communication between the often resource-constrained Industrial IoT (IIoT) devices in these networks must be protected against strong adversaries. While the necessary cryptographic primitives are well-established in the modern Internet, they lead to overhead that is often unacceptable for IIoT devices. Especially, the nearly static processing and bandwidth overhead for message authentication, which is crucial to thwart data manipulation, is challenging in IIoT environments where primarily short messages with tight latency bounds are sent. Meanwhile, precisely this message authentication thwarts the manipulation of sensor readings and control commands that can have detrimental safety and security implications due to the cyber-physical properties of the IIoT.

6.1 Revisting the Research Questions

In this dissertation, we propose various Message Authentication Code (MAC) schemes and protocols to improve the efficiency of message authentication in resource-constrained IIoT environments. We identified three dimensions along which optimizations to message authentication bring major benefits. We captured these three dimensions in three research questions, which we revisit in the following.

RQ1 – How can the bandwidth overhead of message authentication be optimized?

To answer this first research question, we study progressive authentication and MAC aggregation in Chapter 3. Both approaches build upon the idea of combining the

authentication tags of multiple messages to conserve valuable Protocol Data Unit (PDU) space. As a result, received messages may not be immediately authenticated, but the overhead of message authentication is split over multiple messages.

In progressive authentication, received messages are optimistically processed despite initially reduced security. By introducing the sandwich attack, we demonstrate how an adversary can take advantage of current progressive authentication schemes and, in the worst case, covertly circumvent all protection. To mitigate this vulnerability, we design R2D2 to provide provably optimal and quantifiable resilience against the sandwich attack and integrate it into the novel SP-MAC scheme. Our efficient SP-MAC scheme not only protects against the sandwich attack, but also increases resilience to packet loss in general. Thus, progressive authentication remains a promising candidate to ensure message authentication in scenarios where strong security is needed, bandwidth is restricted, and low latency is crucial, even on wireless links.

The design of R2D2 also shows promising results when used as a MAC aggregation scheme. In MAC aggregation, we do not care about optimistic processing but only whether and when a message is fully authenticated. While MAC aggregation is known for over 15 years [115], its potential over wireless and thus lossy channels has not been studied thus far. Contrary to initial intuition, we demonstrate that MAC aggregation can indeed improve authenticated goodput significantly even if up to 10 % of packets are lost. We validate these results by integrating MAC aggregation into DTLS 1.3 and deploying the protocol in a physical testbed.

Overall, we conclude that both progressive authentication and MAC aggregation can significantly reduce the bandwidth overhead of message authentication in constrained environments. This dissertation demonstrates especially that these results also hold for wireless channels where messages may be lost, as long as we choose the right aggregation scheme and parameters for the constraints of a given scenario.

RQ2 – How can cryptographic processing and resulting delays of message authentication be minimized?

In Chapter 4, we task ourselves with reducing the overhead of cryptographic processing without relying on faster processors or hardware acceleration. Such optimizations are crucial in IIoT environments, where devices must remain cost-effective, energy-efficient, and compact. Hence, we explore possibilities to use available computational resources more efficiently.

One flexible approach we explore is RePeL, an efficient library designed to embed authentication data into unused fields of the employed protocols. By eliminating the need for an additional security layer, RePeL not only minimizes bandwidth overhead but also enables retrofitting of security features into legacy systems. Furthermore, RePeL supports offloading cryptographic processing to Bump-in-the-Wire (BitW) devices, providing an additional avenue for efficiency gains.

However, despite RePeL's efficient embedding and verification of authentication data, our evaluation reveals that the computation of authentication tags remains the primary bottleneck to the overall processing latency. To specifically address this

limitation for IIoT scenarios with short messages, we introduce the BP-MAC scheme. This MAC scheme treats each message as a combination of single-bit messages, for which the two possible authentication tags can be precomputed. Authentication and verification are then reduced to selecting and efficiently combining the appropriate precomputed tags. Thereby, BP-MAC demonstrates less overall cryptographic processing for short messages of only a few bytes, which is partially preprocessable.

We thus show that authentication data can be efficiently embedded into and extracted from unused protocol fields in industrial applications, but, in the end, the cryptographic processing of the authentication tags is the actual bottleneck. To fill a gap in available efficient MAC schemes with strong security guarantees, we propose the BP-MAC scheme optimized for the often short messages in IIoT scenarios. Ultimately, combining protocol-level optimizations with selecting a suitable MAC scheme can significantly accelerate cryptographic processing in resource-constrained environments.

RQ3 – How can the efficiency of symmetric cryptography be unlocked for group communication systems?

As a symmetric cryptographic method, MACs offer greater efficiency than their asymmetric counterpart, *i.e.*, digital signatures. However, using symmetric keys among more than two parties introduces the risk of impersonation attacks. As a result, group communication systems typically rely on digital signatures for message authentication. To harness the efficiency of MACs in group communication systems, the TESLA protocol [188, 189] and multiplexing of source-verifying information [44, 187] have been proposed. Yet, these solutions come with significant trade-offs: TESLA introduces verification delays, while source multiplexing incurs additional bandwidth overhead, which are both undesirable for IIoT applications. In Chapter 5, we therefore investigate how the specific characteristics of constrained environments can be exploited to enable the secure use of MACs in group communication systems.

With this dissertation, we unlock the efficiency of symmetric cryptography for two novel scenarios. First, we design Caiba to provide bandwidth-efficient source authentication to multicast communication in bus-based communication systems. Caiba relies on two interleaved authentication tags that authenticate the source and integrity of messages separately. Only if the authenticator removes the correct source-authenticating tag during transmission can the multiple receivers of a message verify its authentication tag. Meanwhile, these receivers cannot impersonate the sender to the authenticator and thus to the entire network. Caiba thus takes advantage of the broadcasting nature of buses and their low speed to realize MAC-based source authentication in multicast communication with neither authentication delay nor bandwidth overhead compared to basic group-key based authentication.

As a second contribution, we design the Madtls protocol. Madtls extends DTLS to enable the secure integration of middleboxes into end-to-end secured communication channels. These middleboxes can be read-only, modify existing packets, drop packets, or inject novel packets into a communication session, while it is tightly controlled which fields can be accessed by which middlebox. Madtls realizes an efficient protocol to integrate nowadays indispensable middleboxes into end-to-end protected

communication channels. Its features accommodate the unique needs of the IIoT, while benefiting from the fact that IIoT networks are usually managed by a single entity, which ensures that non-compromised middleboxes pursue a common goal.

IIoT networks have unique characteristics not necessarily found in the Internet, such as low-speed communication via a CAN bus network, which appears to limit the deployment of cybersecurity solutions. However, by focusing on these particularities, we use them to our advantage to unlock the efficiency of symmetric cryptography and bring lightweight message authentication solutions to group communication systems.

6.2 Future Steps and Research Directions

With the first two research questions, we address the two IIoT constraints most impacted by message authentication: processing and bandwidth overhead. The final research question then deals with carrying these benefits of symmetric cryptography-based message authentication to group communication systems. Throughout all three dimensions, we demonstrate high potential to optimize message authentication for a given scenario. Our work thus lays the foundation for standardizing more efficient protocols and schemes, such as BP-MAC or MAC aggregation in DTLS 1.3, and supports such future efforts with open-source implementations. Standardizing the newly proposed cryptographic schemes and protocols within this dissertation is the next logical step to generate real-world impact, alongside analyzing said methods in large-scale prototypical deployments.

Alongside, further optimizations of the proposed schemes can also be explored. BP-MAC promises to lend itself to efficient hardware implementations, which could enable even faster cryptographic processing. Such a faster BP-MAC implementation could be one enabler for Caiba to leave the lab. After all, a fast authenticator dealing with CAN bandwidth of up to 1 Mbit/s or even the CAN FD protocol is necessary for protecting modern cars.

Especially, the work on Caiba, however, also lays the foundation to explore related research areas in vehicular cybersecurity. The Reactive Bit Flipping (RBF) scheme used by the authenticator risks becoming a powerful tool for adversaries, as it shows potential to covertly overwrite CAN traffic, stealthily disrupt acknowledge and error frame processing, or even enable targeted destruction of specific transceivers. Each of these attacks becomes potentially possible by precisely timing which bits are overwritten. The potential to use RBF offensively as well as devising defense mechanisms against such attacks is thus crucial to stay ahead of criminals.

Furthermore, the impressive optimization potential of MAC aggregation in DTLS 1.3 motivated studying further protocols. Here, protocols such as LoRa [151] and Sigfox [77] that already heavily truncate authentication tags to deal with limited PDU space offer an exciting research avenue. The provided security levels by truncated tags do, however, risk not meeting the requirements towards critical infrastructure that many IIoT scenarios must abide by. Therefore, it should be studied if MAC aggregation allows us to use the available space more efficiently to achieve higher security guarantees for message authentication in LoRa or Sigfox.

Overall, this dissertation lays the foundation for more efficient message authentication in future IIoT protocols. The proposed schemes and protocols expose significant untapped potential to optimize processing and bandwidth overhead of message authentication, and also steer research into promising directions to identify further optimizations in the future.

6.3 Final Remarks

Improving the security in resource-constrained environments is a pressing challenge due to powerful adversaries actively exploiting exposed weaknesses [138]. Some areas to protect these critical infrastructures, such as intrusion detection [133] or lightweight encryption algorithms [37], received significant attention over the last years. Yet, researching efficient message authentication did not receive its deserved and necessary attention, potentially because the mostly static overhead only becomes challenging for short messages as often found in the IIoT.

In this dissertation, we show how message authentication can be significantly improved in these scenarios along the critical dimensions of bandwidth and processing overhead. We also show how the benefits of symmetric cryptography relative to heavier digital signatures can be brought to group communication systems. Our work uncovers significant potential improvements and validates them in simulative evaluations and prototypical deployments to provide high confidence in their results. We thus lay the foundation for a next generation of standards with greatly improved performance for resource-constrained devices and networks, as well as provide impulses for research on further improvements in this domain.

Our work ensures that the performance of MAC schemes keeps pace with advancements across other layers of the ISO/OSI model, as required by modern and future applications. In doing so, we strengthen the security foundations of the IIoT, helping it stay ahead in the ongoing arms race against increasingly capable adversaries, daily threatening our safety and security.

Bibliography

[1] AMD LogiCORE™ CAN IP core . https://www.xilinx.com/products/intellectual-property/do-di-can.html. Last accessed: 12.10.2025.

[2] Arrowhead, Ahead of the Future. http://www.arrowhead.eu. Last Accessed: 12.10.2025.

[3] CAN Injection: Keyless Car Theft. https://kentindell.github.io/2023/04/03/can-injection/. Last accessed: 12.10.2025.

[4] NXP FlexCAN Controller. https://www.nxp.com/products/nxp-product-information/ip-block-licensing/flexcan-controller:FLEXCAN-CONTROLLER. Last accessed: 12.10.2025.

[5] Quickdraw Snort Ruleset. https://github.com/digitalbond/Quickdraw-Snort/blob/master/modbus.rules. Last Accessed: 12.10.2025.

[6] Snort. https://www.snort.org/. Last Accessed: 12.10.2025.

[7] ISO/IEC 9797-1:2011 Information technology - Security techniques - Message Authentication Codes (MACs) - Part 1: Mechanisms using a block cipher, 2011.

[8] OPC Unified Architecture Specification – Part 1: Overview and Concepts. Standard, OPC Foundation, 2017.

[9] MODBUS/TCP Security Protocol Specification. Standard, Modbus Organization, 2018.

[10] Power systems management and associated information exchange - Data and communications security - Part 6: Security for IEC 61850. Technical Report Iec 62351-6:2020, International Electrotechnical Commission, 2020.

[11] Ashfaq Ahmed, Arafat Al-Dweik, Youssef Iraqi, Husameldin Mukhtar, Muhammad Naeem, and Ekram Hossain. Hybrid Automatic Repeat Request (HARQ) in Wireless Communications Systems and Standards: A Contemporary Survey. *IEEE Communications Surveys & Tutorials*, 23(4), 2021.

[12] Taehyun Ahn, Jiwon Kwak, and Seungjoo Kim. mdTLS: How to Make middlebox-aware TLS more efficient? In *Proceedings of the International Conference on Information Security and Cryptology (ICISC'23)*, 2023.

[13] Ayhan Akbas, Huseyin Ugur Yildiz, Bulent Tavli, and Suleyman Uludag. Joint Optimization of Transmission Power Level and Packet Size for WSN Lifetime Maximization. *IEEE Sensors Journal*, 16(12), 2016.

[14] Ismet Aktas, Alexander Bentkus, Florian Bonanati, et al. Position Paper: Wireless Technologies for Industrie 4.0. Technical report, VDE, 2017.

[15] Emad Aliwa, Omer Rana, Charith Perera, and Peter Burnap. Cyberattacks and Countermeasures for In-Vehicle Networks. *ACM Computing Surveys*, 54(1), 2021.

[16] Tejasvi Alladi, Vinay Chamola, and Sherali Zeadally. Industrial Control Systems: Cyberattack Trends and Countermeasures. *Computer Communications*, 155, 2020.

[17] Ralph Ankele, Florian Böhl, and Simon" Friedberger. MergeMAC: a MAC for authentication with strict time constraints and limited bandwidth. In *Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS'18)*, 2018.

[18] Filipe Apolinári, Jo ao Guiomar, Éric Hervé, Sven Hrastnik, Nelson Escravana, Miguel L. Pardal, and Miguel Correia. ComSEC: Secure communications for baggage handling systems. In *Proceedings of the European Symposium on Research in Computer Security Workshops (CPS4CIP'22)*, 2022.

[19] Emekcan Aras, Nicolas Small, Gowri Sankar Ramachandran, Stéphane Delbruel, Wouter Joosen, and Danny Hughes. Selective Jamming of LoRaWAN using Commodity Hardware. In *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous'17)*, 2017.

[20] Frederik Armknecht, Paul Walther, Gene Tsudik, Martin Beck, and Thorsten Strufe. ProMACs: Progressive and Resynchronizing MACs for Continuous Efficient Authentication of Message Streams. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS'20)*, 2020.

[21] Jean-Philippe Aumasson and Daniel J Bernstein. SipHash: A Fast Short-Input PRF. In *Proceedings of the International Conference on Cryptology in India (INDOCRYPT'12)*, 2012.

[22] AUTOSAR. Specification of Secure Onboard Communication Protocol, AUTOSAR FO R20-11. Standard, 2020. `https://www.autosar.org/fileadmin/standards/R20-11/FO/AUTOSAR_PRS_SecOcProtocol.pdf`.

[23] Wallace C. Babcock. Intermodulation interference in radio systems frequency of occurrence and control by channel selection. *The Bell System Technical Journal*, 32(1), 1953.

[24] Christoph Bachhuber, Eckehard Steinbach, Martin Freundl, and Martin Reisslein. On the Minimization of Glass-to-Glass and Glass-to-Algorithm Delay in Video Communication. *IEEE Transactions on Multimedia*, 20(1), 2017.

[25] Amira Barki et al. M2M Security: Challenges and Solutions. *IEEE Communications Surveys & Tutorials*, 18(2), 2016.

[26] Brian Batke, Joakim Wiberg, and Dennis Dubé. CIP security phase 1 secure transport for Ethernet/IP. In *Proceedings of the ODVA Industry Conference*, 2015.

[27] Jan Bauer, René Helmke, Alexander Bothe, and Nils Aschenbruck. CAN't track us: Adaptable privacy for ISOBUS controller area networks. *Computer Standards and Interfaces*, 66, 2019.

[28] Giampaolo Bella, Pietro Biondi, Gianpiero Costantino, and Ilaria Matteucci. TOUCAN: A proTocol tO secUre Controller Area Network. In *Proceedings of the ACM Workshop on Automotive Cybersecurity (AutoSec'19)*, 2019.

[29] Mihir Bellare. New Proofs for NMAC and HMAC: Security Without Collision-Resistance. In *Proceedings of the Annual International Cryptology Conference (CRYPTO'06)*, 2006.

[30] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying Hash Functions for Message Authentication. In *Proceedings of the Annual International Cryptology Conference (CRYPTO'96)*, 1996.

[31] Mihir Bellare, Roch Guérin, and Phillip Rogaway. XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions. In *Proceedings of the Annual International Cryptology Conference (Crypto'95)*, 1995.

[32] Giuseppe Bernieri, Stefano Cecconello, Mauro Conti, and Gianluca Lain. TAMBUS: A novel authentication method through covert channels for securing industrial networks. *Computer Networks*, 183, 2020.

[33] Daniel J Bernstein. Stronger Security Bounds for Wegman-Carter-Shoup Authenticators. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt'05)*, 2005.

[34] Daniel J Bernstein. The Poly1305-AES Message-Authentication Code. In *Proceedings of the International Workshop on Fast Software Encryption (FSE'05)*, 2005.

[35] Laksh Bhatia, Michael Breza, Ramona Marfievici, and Julie A McCann. Dataset: LoED: The LoRaWAN at the Edge Dataset. In *Proceedings of the Third Workshop on Data: Acquisition To Analysis (DATA'20)*, 2020.

[36] Rohit Bhatia, Vireshwar Kumar, Khaled Serag, Z Berkay Celik, Mathias Payer, and Dongyan Xu. Evading Voltage-Based Intrusion Detection on Automotive CAN. In *Proceedings of the Network and Distributed Systems Security Symposium (NDSS'21)*, 2021.

[37] Alex Biryukov and Leo Perrin. State of the Art in Lightweight Symmetric Cryptography. 2017.

[38] John Black and Phillip Rogaway. A block-cipher mode of operation for parallelizable message authentication. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt'02)*, 2002.

[39] Dan Boneh and Victor Shoup. A Graduate Course in Applied Cryptography, 2020.

[40] Pietro Borrello, Andreas Kogler, Martin Schwarzl, Moritz Lipp, Daniel Gruss, and Michael Schwarz. ÆPIC Leak: Architecturally Leaking Uninitialized Data from the Microarchitecture. In *Proceedings of the 31st USENIX Security Symposium (USENIX Sec'22)*, 2022.

[41] An Braeken. Public key versus symmetric key cryptography in client–server authentication protocols. *International Journal of Information Security*, 21(1), 2022.

[42] Ismail Butun, Patrik Österberg, and Houbing Song. Security of the Internet of Things: Vulnerabilities, Attacks, and Countermeasures. *IEEE Communications Surveys & Tutorials*, 22(1), 2020.

[43] Sébastien Canard, Aïda Diop, Nizar Kheir, Marie Paindavoine, and Mohamed Sabt. BlindIDS: Market-Compliant and Privacy-Friendly Intrusion Detection System over Encrypted Traffic. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security (ASIA CCS'17)*, 2017.

[44] Ran Canetti, Juan Garay, Gene Itkis, Daniele Micciancio, Moni Naor, and Benny Pinkas. Multicast Security: A Taxonomy and some Efficient Constructions. In *Proceedings of the 18th Annual Joint Conference on Computer Communications (INFOCOM'99)*, 1999.

[45] B. Carpenter. Architectural Principles of the Internet. RFC 1958, IETF, 1996.

[46] B. Carpenter. Internet Transparency. RFC 2775, IETF, 2000.

[47] B. Carpenter and B. Liu. Limited Domains and Internet Protocols. RFC 8799, IETF, 2020.

[48] Marco Caselli, Emmanuele Zambon, and Frank Kargl. Sequence-aware Intrusion Detection in Industrial Control Systems. In *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security (CPSS'15)*, 2015.

[49] Marco Caselli, Emmanuele Zambon, Jonathan Petit, and Frank Kargl. Modeling Message Sequences for Intrusion Detection in Industrial Control Systems. In *Proceedings of the International Conference on Critical Infrastructure Protection (ICCIP 15)*, 2015.

[50] John Henry Castellanos, Daniele Antonioli, Nils Ole Tippenhauer, and Martín Ochoa. Legacy-Compliant Data Authentication for Industrial control System Traffic. In *Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS'17)*, 2017.

[51] John Henry Castellanos, Daniele Antonioli, Nils Ole Tippenhauer, and Martín Ochoa. Legacy-Compliant Data Authentication for Industrial Control System Traffic. In *Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS'17)*, 2017.

[52] Gianluca Cena, Ivan Cibrario Bertolotti, Tingting Hu, and Adriano Valenzano. On a software-defined CAN controller for embedded systems. *Computer Standards & Interfaces*, 63, 2019.

[53] Fabricio E Rodriguez Cesen, Levente Csikor, Carlos Recalde, Christian Esteve Rothenberg, and Gergely Pongrácz. Towards Low Latency Industrial Robot Control in Programmable Data Planes. In *Proceedings of the Conference on Network Softwarization (NetSoft'20)*, 2020.

[54] Yacine Challal, Hatem Bettahar, and Abdelmadjid Bouabdallah. A taxonomy of multicast data origin authentication: Issues and solutions. *IEEE Communications Surveys & Tutorials*, 6(3), 2004.

[55] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, Tadayoshi Kohno, et al. Comprehensive experimental analyses of automotive attack surfaces. In *Proceedings of the 20th USENIX Security Symposium (USENIX Sec'11)*, 2011.

[56] Yuang Chen and Thomas Kunz. Performance Evaluation of IoT Protocols under a Constrained Wireless Access Network. In *Proceedings of International Conferemce on Selected Topics in Mobile & Wireless Networking (MoWNeT'16)*, 2016.

[57] Kyong-Tak Cho and Kang G Shin. Fingerprinting Electronic Control Units for Vehicle Intrusion Detection. In *Proceedings of the 25th USENIX Security Symposium (USENIX Sec'16')*, 2016.

[58] Kyong-Tak Cho and Kang G Shin. Viden: Attacker Identification on In-Vehicle Networks. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS'17)*, 2017.

[59] Wonsuk Choi, Kyungho Joo, Hyo Jin Jo, Moon Chan Park, and Dong Hoon Lee. VoltageIDS: Low-Level Communication Characteristics for Automotive Intrusion Detection System. *IEEE Transactions on Information Forensics and Security*, 13(8), 2018.

[60] Carlos Alexandre Gouvea Da Silva and Carlos Marcelo Pedroso. MAC-Layer Packet Loss Models for Wi-Fi Networks: A Survey. *IEEE Access*, 7, 2019.

[61] Markus Dahlmanns, Johannes Lohmöller, Ina Berenice Fink, Jan Pennekamp, Klaus Wehrle, and Martin Henze. Easing the Conscience with OPC UA: An Internet-Wide Study on Insecure Deployments. In *Proceedings of the ACM Internet Measurement Conference (IMC'20)*, 2020.

[62] Markus Dahlmanns, Johannes Lohmöller, Jan Pennekamp, Jörn Bodenhausen, Klaus Wehrle, and Martin Henze. Missed Opportunities: Measuring the Untapped TLS Support in the Industrial Internet of Things. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security (AsiaCCS'22)*, 2022.

[63] Xavier de Carné de Carnavalet and Paul C. van Oorschot. A Survey and Analysis of TLS Interception Mechanisms and Motivations: Exploring How End-to-End TLS is Made "End-to-Me" for Web Traffic. *ACM Computing Surveys*, 55(13s), 2023.

[64] Alvise de Faveri Tron, Stefano Longari, Michele Carminati, Mario Polino, and Stefano Zanero. CANflict: Exploiting Peripheral Conflicts for Data-Link Layer Attacks on Automotive Networks. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS'22)*, 2022.

[65] Tim Dierks and Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, IETF, 2008.

[66] Henry Dol, Koen Blom, Dimitri Sotnik, Ivor Nissen, Roald Otnes, Arwid Komulainen, and Filippo Campagnaro. EDA-SALSA: Development of a self-reconfigurable protocol stack for robust underwater acoustic networking. In *IEEE/MTS Oceans*, 2023.

[67] D. Dolev and A. Yao. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, 29(2), 1983.

[68] Huayi Duan, Cong Wang, Xingliang Yuan, Yajin Zhou, Qian Wang, and Kui Ren. LightBox: Full-Stack Protected Stateful Middlebox at Lightning Speed. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS'19)*, 2019.

[69] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the 29th international Conference on Local Computer Networks (LCN'04)*, 2004.

[70] D. Eastlake 3rd. US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF). RFC 6234, IETF, 2011.

[71] Oliver Eikemeier, Marc Fischlin, Jens-Fabian Götzmann, Anja Lehmann, Dominique Schröder, Peter Schröder, and Daniel Wagner. History-Free Aggregate Message Authentication Codes. In *Proceedings of the International Conference on Security and Cryptography for Networks (SCN'10)*, 2010.

[72] Ertem Esiner, Daisuke Mashima, Binbin Chen, Zbigniew Kalbarczyk, and David Nicol. F-Pro: A Fast and Flexible Provenance-Aware Message Authentication Scheme for Smart Grid. In *Proceedings of the IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm'19)*, 2019.

[73] Ertem Esiner, Utku Tefek, Daisuke Mashima, Binbin Chen, Zbigniew Kalbarczyk, and David M Nicol. Message Authentication and Provenance Verification for Industrial Control Systems. *ACM Transactions on Cyber-Physical Systems*, 7(4), 2023.

[74] Sharanya Eswaran, James Edwards, Archan Misra, and Thomas F. La Porta. Adaptive In-Network Processing for Bandwidth and Energy Constrained Mission-Oriented Multihop Wireless Networks. *IEEE Transactions on Mobile Computing*, 11(9), 2012.

[75] Federal Office for Information Security. Technical Guideline – Cryptographic Algorithms and Key Lengths. Technical Report BSI TR-02102-1, 2024.

[76] Marc Fischlin. Stealth Key Exchange and Confined Access to the Record Protocol Data in TLS 1.3. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS'23)*, 2023.

[77] Florian Euchner, Felix Fellhauer, Marx Gauger. Sigfox Radio Protocol Overview and Specifications. Technical report, 2018.

[78] European Committee for Electrotechnical Standardization. Industrial communications subsystem based on ISO 11898(CAN) for controller-device interfaces – Part 4: CANopen. Standard, Cenelec, 2002.

[79] Mahsa Foruhandeh, Yanmao Man, Ryan Gerdes, Ming Li, and Thidapat Chantem. SIMPLE: single-frame based physical layer identification for intrusion detection and prevention on in-vehicle networks. In *Proceedings of the 35th Annual Computer Security Applications Conference (ACSAC'19)*, 2019.

[80] Ian Foster, Andrew Prudhomme, Karl Koscher, and Stefan Savage. Fast and Vulnerable: A Story of Telematic Failures. In *Proceedings of the 9th USENIX Workshop on Offensive Technologies (WOOT'15)*, 2015.

[81] Frenzel+Berg. CANopen Chip CO4011. `https://www.frenzel-berg.de/fileadmin/FrenzelBerg/Datenblaetter/CANopen_Chip/ds_co4011b_en.pdf`. Last Accessed: 12.10.2025.

[82] Brendan Galloway and Gerhard P Hancke. Introduction to Industrial Control Networks. *IEEE Communications Surveys & Tutorials*, 15(2), 2012.

[83] Peter Gaži, Krzysztof Pietrzak, and Michal Rybár. The exact PRF-security of NMAC and HMAC. In *Proceedings of the 34th Annual Cryptology Conference (CRYPTO'14)*, 2014.

[84] Benjamin Glas, Jorge Guajardo, Hamit Hacioglu, Markus Ihle, Karsten Wehefritz, and Attila Yavuz. Signal-based Automotive Communication Security and Its Interplay with Safety Requirements. In *Proceedings of the 10th International Conference on Embedded Security in Cars (ESCAR'12)*, 2012.

[85] Michael Goetz and Ivor Nissen. GUWMANET — Multicast routing in Underwater Acoustic Networks. In *Proceedings of the 2012 Military Communications and Information Systems Conference (MCC'12)*, 2012.

[86] Jonathan Goh, Sridhar Adepu, Khurum Nazir Junejo, and Aditya Mathur. A Dataset to Support Research in the Design of Secure Water Treatment Systems. In *Proceedings of International Conference on Critical Information Infrastructures Security (CRITIS'16)*, 2016.

[87] Zheng Gong, Pieter Hartel, Svetla Nikova, Shao-Hua Tang, and Bo Zhu. TuLP: A Family of Lightweight Message Authentication Codes for Body Sensor Networks. *Journal of Computer Science and Technology*, 29(1), 2014.

[88] John Griffith. Learn the inner workings of a CAN bus driver and how to debug your system. Technical article, Texas Instruments, 2016. `https://www.ti.com/lit/ta/ssztbo8/ssztbo8.pdf`.

[89] E. Grossman. Deterministic Networking Use Cases. RFC 8578, IETF, 2019.

[90] Bogdan Groza, Stefan Murvay, Anthony Van Herrewege, and Ingrid Verbauwhede. LiBrA-CAN: A Lightweight Broadcast Authentication Protocol for Controller Area Networks. In *Proceedings of the International Conference on Cryptology and Network Security (CANS'12)*, 2012.

[91] Bogdan Groza, Lucian Popa, and Pal-Stefan Murvay. Canto-covert authentication with timing channels over optimized traffic flows for can. *IEEE Transactions on Information Forensics and Security*, 16, 2020.

[92] Bogdan Groza, Lucian Popa, and Pal-Stefan Murvay. Highly Efficient Authentication for CAN by Identifier Reallocation With Ordered CMACs. *IEEE Transactions on Vehicular Technology*, 69(6), 2020.

[93] Paul Grubbs, Arasu Arun, Ye Zhang, Joseph Bonneau, and Michael Walfish. Zero-Knowledge Middleboxes. In *Proceedings of the 31st USENIX Security Symposium (USENIX Sec'22)*, 2022.

[94] Shay Gueron. Memory Encryption for General-Purpose Processors. *IEEE Security & Privacy*, 14(6), 2016.

[95] Csaba Györgyi, Károly Kecskeméti, Péter Vörös, Géza Szabó, and Sándor Laki. In-network Solution for Network Traffic Reduction in Industrial Data Communication. In *Proceedings of the International Conference on Network Softwarization (NetSoft'21)*, 2021.

[96] Csaba Györgyi, Péter Vörös, Károly Kecskeméti, Géza Szabó, and Sándor Laki. Adaptive Network Traffic Reduction on the Fly With Programmable Data Planes. *IEEE Access*, 11, 2023.

[97] Taieb Hamza, Georges Kaddoum, Aref Meddeb, and Georges Matar. A survey on intelligent MAC layer jamming attacks and countermeasures in WSNs. In *Proceedings of the 84th Vehicular Technology Conference (VTC-Fall'16)*, 2016.

[98] Juhyeng Han, Seongmin Kim, Jaehyeong Ha, and Dongsu Han. SGX-Box: Enabling Visibility on Encrypted Traffic using a Secure Middlebox Module. In *Proceedings of the First Asia-Pacific Workshop on Networking (APNET'17)*, 2017.

[99] Thomas Hänel, Leonhard Brüggemann, Felix Loske, and Nils 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'21)*, 2021.

[100] Ahmed Hazem and HA Fahmy. LCAP-A Lightweight CAN Authentication Protocol for Securing In-Vehicle Networks. In *Proceedings of the 10th International Conference on Embedded Security in Cars (ESCAR'12)*, 2012.

[101] Jens Hiller, Martin Henze, Martin Serror, Eric Wagner, Jan Niklas Richter, and Klaus Wehrle. Secure Low Latency Communication for Constrained Industrial IoT Scenarios. In *Proceedings of the 42nd IEEE Conference on Local Computer Networks (LCN'18)*, 2018.

[102] Jens Hiller, Jan Pennekamp, Markus Dahlmanns, Martin Henze, Andriy Panchenko, and Klaus Wehrle. Tailoring Onion Routing to the Internet of Things: Security and Privacy in Untrusted Environments. In *Proceedings of the 27th International Conference on Network Protocols (ICNP'19)*, 2019.

[103] Shoichi Hirose and Junji Shikata. Non-adaptive Group-Testing Aggregate MAC Scheme. In *Proceedings of the International Conference on Information Security Practice and Experience (ISPEC'18)*, 2018.

[104] HMS Networks. HMS Networks – fieldbus, industrial Ethernet and wireless, 2022.

[105] J. Hong, J.-G. Hong, X. de Foy, M. Kovatsch, E. Schooler, and D. Kutscher. Internet of Things (IoT) Edge Challenges and Functions. RFC 9556, IETF, 2024.

[106] Yao-win Hong, Wan-jen Huang, Fu-hsuan Chiu, and C.-c. Jay Kuo. Cooperative Communications in Resource-Constrained Wireless Networks. *IEEE Signal Processing Magazine*, 24(3), 2007.

[107] Abdulmalik Humayed, Jingqiang Lin, Fengjun Li, and Bo Luo. Cyber-Physical Systems Security–A Survey. *IEEE Internet of Things Journal*, 4(6), 2017.

[108] International Organization for Standardization. Road vehicles - Controller area network (CAN) – Part 3: Low-speed, fault-tolerant, medium-dependent interface. ISO 11898-3:2006, 2006.

[109] International Organization for Standardization. Road vehicles - FlexRay communications system – Part 1: General information and use case definition. ISO 17458-{1}:2013, 2013.

[110] International Organization for Standardization. Road vehicles - Controller area network (CAN) – Part 1: Data link layer and physical signalling. ISO 11898-1:2015, 2015.

[111] International Organization for Standardization. Road vehicles - Local Internconnect Network (LIN) – Part 1: General information and use case definition. ISO 17987-1:2016, 2016.

[112] International Organization for Standardization. Road vehicles - Controller area network (CAN) – Part 2: High-speed medium access unit. ISO 11898-2:2022, 2022.

[113] Hyo Jin Jo, Jin Hyun Kim, Hyon-Young Choi, Wonsuk Choi, Dong Hoon Lee, and Insup Lee. MAuth-CAN: Masquerade-Attack-Proof Authentication for In-Vehicle Networks. *IEEE Transactions on Vehicular Technology*, 69(2), 2020.

[114] Sotirios Katsikeas, Konstantinos Fysarakis, Andreas Miaoudakis, Amaury Van Bemten, Ioannis Askoxylakis, Ioannis Papaefstathiou, and Anargyros Plemenos. Lightweight & Secure Industrial IoT Communications via the MQ Telemetry Transport Protocol. In *Proceedings of the IEEE Symposium on Computers and Communications (ISCC'17)*, 2017.

[115] Jonathan Katz and Andrew Y Lindell. Aggregate Message Authentication Codes. In *Proceedings of the Cryptographers' Track at the RSA Conference*, 2008.

[116] S Kelly and S Frankel. Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec. RFC 4868, IETF, 2007.

[117] Shiho Kim and Rakesh Shrestha. AUTOSAR Embedded Security in Vehicles. In *Automotive Cyber Security: Introduction, Challenges, and Standardization*. 2020.

[118] Marcel Kneib and Christopher Huth. Scission: Signal characteristic-based sender identification and intrusion detection in automotive networks. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS'18)*, 2018.

[119] Marcel Kneib, Oleg Schell, and Christopher Huth. EASI: Edge-Based Sender Identification on Resource-Constrained Platforms for Automotive Networks. In *Proceedings of the Network and Distributed System Security Symposium (NDSS'20)*, 2020.

[120] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M Frans Kaashoek. The Click Modular Router. *Transactions on Computer Systems*, 18(3), 2000.

[121] Thomas Kohler, Ruben Mayer, Frank Dürr, Marius Maaß, Sukanya Bhowmik, and Kurt Rothermel. P4CEP: Towards In-Network Complex Event Processing. In *Proceedings of the Morning Workshop on In-Network Computing (NetCompute'18)*, 2018.

[122] Vladimir Kolesnikov. MAC Aggregation with Message Multiplicity. In *Proceedings of the International Conference on Security and Cryptography for Networks (SCN'12)*, 2012.

[123] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage. Experimental Security Analysis of a Modern Automobile. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'10)*, 2010.

[124] Shaguftha Zuveria Kottur, Krishna Kadiyala, Praveen Tammana, and Rinku Shah. Implementing ChaCha Based Crypto Primitives on Programmable SmartNICs. In *Proceedings of the ACM SIGCOMM Workshop on Formal Foundations and Security of Programmable Network Infrastructures*, 2022.

[125] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. HMAC: Keyed-Hashing for Message Authentication. RFC 2104, IETF, 1997.

[126] Ted Krovetz. Message Authentication on 64-bit Architectures. In *Proceedings of the International Workshop on Selected Areas in Cryptography (SAC'06)*, 2006.

[127] Ted Krovetz et al. UMAC: Message authentication code using universal hashing, 2006. RFC 4418.

[128] Ralf Kundel, Fridolin Siegmund, Jeremias Blendin, Amr Rizk, and Boris Koldehofe. P4STA: High Performance Packet Timestamping with Programmable Packet Processors. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS'20)*, 2020.

[129] Ike Kunze, René Glebke, Jan Scheiper, Matthias Bodenbenner, Robert H Schmitt, and Klaus Wehrle. Investigating the Applicability of In-Network Computing to Industrial Scenarios. In *Proceedings of the 4th International Conference on Industrial Cyber-Physical Systems (ICPS'21)*, 2021.

[130] Ike Kunze, Philipp Niemietz, Liam Tirpitz, René Glebke, Daniel Trauth, Thomas Bergs, and Klaus Wehrle. Detecting Out-Of-Control Sensor Signals in Sheet Metal Forming Using In-Network Computing. In *Proceedings of the 2021 IEEE International Symposium on Industrial Electronics (ISIE'21)*, 2021.

[131] Ryo Kurachi, Yutaka Matsubara, Hiroaki Takada, Naoki Adachi, Yukihiro Miyashita, and Satoshi Horihata. CaCAN-Centralized Authentication System in CAN (Controller Area Network). In *Proceedings of the 14th International Conference on Embedded Security in Cars (ESCAR'14)*, 2014.

[132] Sinan Kurt, Huseyin Ugur Yildiz, Melike Yigit, Bulent Tavli, and Vehbi Cagri Gungor. Packet Size Optimization in Wireless Sensor Networks for Smart Grid Applications. *IEEE Transactions on Industrial Electronics*, 64(3), 2016.

[133] Olav Lamberts, Konrad Wolsing, Eric Wagner, Jan Pennekamp, Jan Bauer, Klaus Wehrle, and Martin Henze. SoK: Evaluations in Industrial Intrusion Detection Research. *Journal of Systems Research*, 3(1), 2023.

[134] Chang Lan, Justine Sherry, Raluca Ada Popa, Sylvia Ratnasamy, and Zhi Liu. Embark: Securely Outsourcing Middleboxes to the Cloud. In *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI'16)*, 2016.

[135] Mihai T Lazarescu. Design of a WSN Platform for Long-term Environmental Monitoring for IoT Applications. *IEEE Journal on emerging and selected topics in circuits and systems*, 3(1), 2013.

[136] Hyunwoo Lee, Zach Smith, Junghwan Lim, Gyeongjae Choi, Selin Chun, Taejoong Chung, and Ted Taekyoung Kwon. maTLS: How to Make TLS middlebox-aware? In *Proceedings of the Network and Distributed System Security Symposium (NDSS'19)*, 2019.

[137] Yousik Lee, Yong-Eun Kim, Jin-Gyun Chung, and Samuel Woo. Real Time Perfect Bit Modification Attack on In-Vehicle CAN. *IEEE Transactions on Vehicular Technology*, 72(12), 2023.

[138] Ted G Lewis. *Critical Infrastructure Protection in Homeland Security: Defending a Networked Nation*. 2019.

[139] He Li, Vireshwar Kumar, Jung-Min Park, and Yaling Yang. Cumulative Message Authentication Codes for Resource-Constrained IoT Networks. *IEEE Internet of Things Journal*, 8(15), 2021.

[140] He Li, Vireshwar Kumar, Jung-Min Jerry Park, and Yaling Yang. Cumulative Message Authentication Codes for Resource-Constrained Networks. In *Proceedings of the 2020 IEEE Conference on Communications and Network Security (CNS'20)*, 2020.

[141] Jian-Qiang Li, F Richard Yu, Genqiang Deng, Chengwen Luo, Zhong Ming, and Qiao Yan. Industrial internet: A survey on the enabling technologies, applications, and challenges. *IEEE Communications Surveys & Tutorials*, 19(3), 2017.

[142] Jie Li, Rongmao Chen, Jinshu Su, Xinyi Huang, and Xiaofeng Wang. ME-TLS: Middlebox-enhanced TLS for Internet-of-Things Devices. *IEEE Internet of Things Journal*, 7(2), 2019.

[143] Xiaozhou Li, Raghav Sethi, Michael Kaminsky, David G Andersen, and Michael J Freedman. Be Fast, Cheap and in Control with SwitchKV. In *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI'16)*, 2016.

[144] Zhengnan Li, Mandar Chitre, and Milica Stojanovic. Underwater Acoustic Communications. *Nature Reviews Electrical Engineering*, 2(2), 2024.

[145] Athanasios Liatifis, Panagiotis Sarigiannidis, Vasileios Argyriou, and Thomas Lagkas. Advancing SDN from OpenFlow to P4: A Survey. *ACM Computing Surveys*, 55(9), 2023.

[146] Chih-Yuan Lin and Simin Nadjm-Tehrani. Timing Patterns and Correlations in Spontaneous SCADA Traffic for Anomaly Detection. In *Proceedings of the International Symposium on Research in Attacks, Intrusions and Defenses (RAID'19)*, 2019.

[147] Chung-Wei Lin and Alberto Sangiovanni-Vincentelli. Cyber-Security for the Controller Area Network (CAN) Communication Protocol. In *Proceedings of the International Conference on Cyber Security (CyberSecurity'12)*, 2012.

[148] Aejaz Nazir Lone, Suhel Mustajab, and Mahfooz Alam. A comprehensive study on cybersecurity challenges and opportunities in the IoT world. *Security and Privacy*, 6(6), 2023.

[149] Stefano Longari, Matteo Penco, Michele Carminati, and Stefano Zanero. CopyCAN: An Error-Handling Protocol based Intrusion Detection System for Controller Area Network. In *Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy*, 2019.

[150] Stefano Longari, Carlo Alberto Pozzoli, Alessandro Nichelini, Michele Carminati, and Stefano Zanero. CANdito: Improving Payload-Based Detection of Attacks on Controller Area Networks. In *Proceedings of the International Symposium on Cyber Security, Cryptology, and Machine Learning (CSCML'25)*, 2023.

[151] LoRa Alliance. LoRaWAN™ 1.1 Specification. Technical report, 2017.

[152] Alessandro Lotto, Francesco Marchiori, Alessandro Brighente, and Mauro Conti. A Survey and Comparative Analysis of Security Properties of CAN Authentication Protocols, 2025.

[153] Zhaojun Lu, Qian Wang, Xi Chen, Gang Qu, Yongqiang Lyu, and Zhenglin Liu. LEAP: A Lightweight Encryption and Authentication Protocol for In-Vehicle Communications. In *Proceedings of the IEEE Intelligent Transportation Systems Conference (ITSC'19)*, 2019.

[154] Mark Luk, Ghita Mezzour, Adrian Perrig, and Virgil Gligor. MiniSec: a secure sensor network communication architecture. In *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks (IPSN'07)*, 2007.

[155] Michele Luvisotto, Zhibo Pang, and Dacfey Dzung. Ultra High Performance Wireless Control for Critical Applications: Challenges and Directions. *IEEE Transactions on Industrial Informatics*, 13(3), 2016.

[156] Atul Luykx, Bart Preneel, Elmar Tischhauser, and Kan Yasuda. A MAC Mode for Lightweight Block Ciphers. In *Proceedings of the International Conference on Fast Software Encryption (FSE'16)*, 2016.

[157] Yangdi Lyu and Prabhat Mishra. A survey of side-channel attacks on caches and countermeasures. *Journal of Hardware and Systems Security*, 2(1), 2018.

[158] Tianle Mai, Haipeng Yao, Song Guo, and Yunjie Liu. In-Network Computing Powered Mobile Edge: Toward High Performance Industrial IoT. *IEEE Network*, 35(1), 2020.

[159] Elia Oronzo Marasco and Francesco Quaglia. AuthentiCAN: a Protocol for Improved Security over CAN. In *Proceedings of the Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4'20)*, 2020.

[160] Daisuke Mashima, Ramkumar Rajendran, Toby Zhou, Binbin Chen, and Biplab Sikdar. On optimization of command-delaying for advanced command authentication in smart grid systems. In *Proceedings of the 2019 IEEE Innovative Smart Grid Technologies-Asia (ISGT Asia'19)*, 2019.

[161] Richard May, Christian Biermann, Jacob Krüger, and Thomas Leich. Asking Security Practitioners: Did You Find the Vulnerable (Mis)Configuration? In *Proceedings of the 19th International Working Conference on Variability Modelling of Software-Intensive Systems*, 2025.

[162] Alan J Michaels, Venkata Sai Srikar Palukuru, Michael J Fletcher, Chris Henshaw, Steven Williams, Thomas Krauss, James Lawlis, and John J Moore. CAN Bus Message Authentication via Co-Channel RF Watermark. *IEEE Transactions on Vehicular Technology*, 71(4), 2022.

[163] Charlie Miller and Chris Valasek. Remote Exploitation of an Unaltered Passenger Vehicle. *Black Hat USA*, 2015(S 91), 2015. `https://illmatics.com/Remote%20Car%20Hacking.pdf`.

[164] Sadia Moriam, Elke Franz, Paul Walther, Akash Kumar, Thorsten Strufe, and Gerhard Fettweis. Protecting Communication in Many-Core Systems against Active Attackers. In *Proceedings of the Great Lakes Symposium on VLSI (GLSVLSI'18)*, 2018.

[165] Nicky Mouha, Bart Mennink, Anthony Van Herrewege, Dai Watanabe, Bart Preneel, and Ingrid Verbauwhede. Chaskey: an efficient MAC algorithm for 32-bit microcontrollers. In *Proceedings of the International Conference on Selected Areas in Cryptography (SAC'14)*, 2014.

[166] Francois Mouton, Louise Leenen, and Hein S Venter. Social Engineering Attack Examples, Templates and Scenarios. *Computers & Security*, 59, 2016.

[167] Multicomp Pro. 4 Channel Digital Storage Oscilloscope. `https://www.farnell.com/datasheets/3155232.pdf`. Last Accessed: 12.10.2025.

[168] Steven J Murdoch and Stephen Lewis. Embedding Covert Channels into TCP/IP. In *International Workshop on Information Hiding*, 2005.

[169] Kit Murdock, David Oswald, Flavio D. Garcia, Jo Van Bulck, Daniel Gruss, and Frank Piessens. Plundervolt: Software-based Fault Injection Attacks against Intel SGX. In *Proceedings of the 41st IEEE Symposium on Security and Privacy (S&P'20)*, 2020.

[170] Michael Müter and Naim Asaj. Entropy-Based Anomaly Detection for In-Vehicle Networks. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV'11)*, 2011.

[171] National Institute of Standards and Technology. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. Technical report, 2005.

[172] National Institute of Standards and Technology. Recommendation for Applications Using Approved Hash Algorithms. Technical report, 2012.

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

[174] National Institute of Standards and Technology. Data Confidentiality: Identifying and Protecting Assets Against Data Breaches. *NIST Special Publication 1800-28*, 2024.

[175] David Naylor, Richard Li, Christos Gkantsidis, Thomas Karagiannis, and Peter Steenkiste. And Then There Were More: Secure Communication for More Than Two Parties. In *Proceedings of the International Conference on emerging Networking EXperiments and Technologies (CoNEXT'17)*, 2017.

[176] David Naylor, Kyle Schomp, Matteo Varvello, Ilias Leontiadis, Jeremy Blackburn, Diego R López, Konstantina Papagiannaki, Pablo Rodriguez Rodriguez, and Peter Steenkiste. Multi-Context TLS (mcTLS) Enabling Secure In-Network Functionality in TLS. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication (SIGCOMM'15)*, 2015.

[177] Alessandro Nichelini, Carlo Alberto Pozzoli, Stefano Longari, Michele Carminati, and Stefano Zanero. CANova: A hybrid intrusion detection framework based on automatic signal classification for CAN. *Computers & Security*, 128, 2023.

[178] Sen Nie, Ling Liu, and Yuefeng Du. Free-Fall: Hacking Tesla from Wireless to CAN Bus. *Black Hat USA*, 25(1), 2017. https://www.blackhat.com/docs/us-17/thursday/us-17-Nie-Free-Fall-Hacking-Tesla-From-Wireless-To-CAN-Bus-wp.pdf.

[179] Sen Nie, Ling Liu, Yuefeng Du, and Wenkai Zhang. Over-the-Air: How we Remotely Compromised the Gateway, BCM, and Autopilot ECUs of Tesla Cars. *Black Hat USA*, 91, 2018. https://i.blackhat.com/us-18/Thu-August-9/us-18-Liu-Over-The-Air-How-We-Remotely-Compromised-The-Gateway-Bcm-And-Autopilot-Ecus-Of-Tesla-Cars-wp.pdf.

[180] Dennis K Nilsson, Ulf E Larson, and Erland Jonsson. Efficient In-Vehicle Delayed Data Authentication Based on Compound Message Authentication Codes. In *Proceedings of the 68th Vehicular Technology Conference (VTC-Fall'08)*, 2008.

[181] Jianting Ning, Geong Sen Poh, Jia-Ch'ng Loh, Jason Chia, and Ee-Chien Chang. PrivDPI: Privacy-Preserving Encrypted Traffic Inspection with Reusable Obfuscated Rules. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS'19)*, 2019.

[182] Y. Nir. ChaCha20, Poly1305, and Their Use in the Internet Key Exchange Protocol (IKE) and IPsec. RFC 7634, IETF, 2015.

[183] Stefan Nürnberger and Christian Rossow. –vatiCAN–Vetted, Authenticated CAN Bus. In *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems (CHES'16)*, 2016.

[184] Franco Oberti, Alessandro Savino, Ernesto Sanchez, Paolo Casasso, Filippo Parisi, and Stefano Di Carlo. CAN-MM: Multiplexed Message Authentication Code for Controller Area Network Message Authentication in Road Vehicles. *IEEE Transactions on Vehicular Technology*, 73(10), 2024.

[185] Xinru Page, Paritosh Bahirat, Muhammad I Safi, Bart P Knijnenburg, and Pamela Wisniewski. The Internet of What? Understanding Differences in Perceptions and Adoption for the Internet of Things. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(4), 2018.

[186] Gennady Pekhimenko, Chuanxiong Guo, Myeongjae Jeon, Peng Huang, and Lidong Zhou. TerseCades: Efficient Data Compression in Stream Processing. In *Proceedings of the USENIX Annual Technical Conference (ATC'18)*, 2018.

[187] Adrian Perrig. The BiBa One-Time Signature and Broadcast Authentication Protocol. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CSS'01)*, 2001.

[188] Adrian Perrig, Ran Canetti, Dawn Song, and J Doug Tygar. Efficient and Secure Source Authentication for Multicast. In *Proceedings of the Network and Distributed Systems Security Symposium (NDSS'01)*, 2001.

[189] Adrian Perrig, Ran Canetti, J Doug Tygar, and Dawn Song. Efficient Authentication and Signing of Multicast Streams over Lossy Channels. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'00)*, 2000.

[190] Mert D Pesé, Jay W Schauer, Junhui Li, and Kang G Shin. S2-CAN: Sufficiently Secure Controller Area Network. In *Proceedings of the 37th Annual Computer Security Applications Conference (ACSAC'21)*, 2021.

[191] Philips Semiconductors. TJA1050 - High speed CAN transceiver. Technical report, 2002.

[192] Rishabh Poddar, Chang Lan, Raluca Ada Popa, and Sylvia Ratnasamy. SafeBricks: Shielding Network Functions in the Cloud. In *Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI'18)*, 2018.

[193] B. Preneel and P.C. van Oorschot. On the Security of Iterated Message Authentication Codes. *IEEE Transactions on Information Theory*, 45(1), 1999.

[194] Bart Preneel and Paul C. Van Oorschot. MDx-MAC and building fast MACs from hash functions. In *Proceedings of the Annual International Cryptology Conference (CRYPTO'95)*, 1995.

[195] S Pushpalatha and KS 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*. 2019.

[196] Michael Rademacher, Hendrik Linka, Thorsten Horstmann, and Martin Henze. Path Loss in Urban LoRa Networks: A Large-Scale Measurement Study. In *Proceedings of the 94th Vehicular Technology Conference (VTC-Fall'21)*. IEEE, 2021.

[197] Andreea-Ina Radu and Flavio D Garcia. LeiA: A lightweight authentication protocol for CAN. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS'16)*, 2016.

[198] Rizwan Hamid Randhawa, Abdul Hameed, and Adnan Noor Mian. Energy efficient cross-layer approach for object security of CoAP for IoT devices. *Ad Hoc Networks*, 92, 2019.

[199] Shahid Raza, Daniele Trabalza, and Thiemo Voigt. 6LoWPAN Compressed DTLS for CoAP. In *Proceedings of the 8th International Conference on Distributed Computing in Sensor Systems (DCOSS'12)*, 2012.

[200] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, IETF, 2018.

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

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

[203] Francesca Righetti, Carlo Vallati, Daniela Comola, and Giuseppe Anastasi. Performance Measurements of IEEE 802.15. 4g Wireless Networks. In *Proceedings of the International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM'19)*, 2019.

[204] R. Rivers. The MD5 Message-Digest Algorithm. RFC 1321, IETF, 1992.

[205] Fabricio Rodriguez, Christian Esteve Rothenberg, and Gergely Pongrácz. In-Network P4-based Low Latency Robot Arm Control. In *Proceedings of the 15th International Conference on emerging Networking EXperiments and Technologies (CoNEXT'22)*, 2019.

[206] Matthew Rogers, Phillip Weigand, Jassim Happa, and Kasper Rasmussen. Detecting CAN Attacks on J1939 and NMEA 2000 Networks. *IEEE Transactions on Dependable and Secure Computing*, 20(3), 2022.

[207] Martin Rosso, Luca Allodi, Emmanuele Zambon, and Jerry den Hartog. A Methodology to Measure the "Cost" of CPS Attacks: Not all CPS Networks are Created Equal. In *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 2024.

[208] Lucca Ruhland, Mari Schmidt, Jan Bauer, and Elmar Padilla. Keeping the Baddies Out and the Bridge Calm: Embedded Authentication for Maritime Networks. In *Proceedings of the 2022 International Symposium on Networks, Computers and Communications (ISNCC'22)*, 2022.

[209] Jan Rüth, René Glebke, Klaus Wehrle, Vedad Causevic, and Sandra Hirche. Towards In-Network Industrial Feedback Control. In *Proceedings of the Morning Workshop on In-Network Computing (NetCompute'18)*, 2018.

[210] Sang Uk Sagong, Xuhang Ying, Andrew Clark, Linda Bushnell, and Radha Poovendran. Cloaking the Clock: Emulating Clock Skew in Controller Area Networks. In *Proceedings of the 9th International Conference on Cyber-Physical Systems (ICCPS'18)*, 2018.

[211] Yore Sankarasubramaniam, Ian F Akyildiz, and SW McLaughlin. Energy efficiency based packet size optimization in wireless sensor networks. In *Proceedings of the First International Workshop on Sensor Network Protocols and Applications (SNPA'03)*, 2003.

[212] Amedeo Sapio, Ibrahim Abdelaziz, Abdulla Aldilaijan, Marco Canini, and Panos Kalnis. In-Network Computation Is a Dumb Idea Whose Time Has Come. In *Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets'17)*, 2017.

[213] Syahril Ramadhan Saufi, Zair Asrar Bin Ahmad, Mohd Salman Leong, and Meng Hee Lim. Challenges and Opportunities of Deep Learning Models for Machinery Fault Detection and Diagnosis: A Review. *IEEE Access*, 7, 2019.

[214] Oleg Schell and Marcel Kneib. SPARTA: Signal Propagation-based Attack Recognition and Threat Avoidance for Automotive Networks. In *Proceedings of the ACM Asia Conference on Computer and Communications Security (AsiaCCS'23)*, 2023.

[215] Eryk Schiller, Andy Aidoo, Jara Fuhrer, Jonathan Stahl, Michael Ziörjen, and Burkhard Stiller. Landscape of IoT security. *Computer Science Review*, 44, 2022.

[216] Jackson Schmandt et al. Mini-MAC: Raising the bar for vehicular security with a lightweight message authentication protocol. *Vehicular Communications*, 9, 2017.

[217] Hendrik Schweppe, Yves Roudier, Benjamin Weyl, Ludovic Apvrille, and Dirk Scheuermann. Car2X Communication: Securing the Last Meter - A Cost-Effective Approach for Ensuring Trust in Car2X Applications Using In-Vehicle Symmetric Cryptography. In *Proceedings of the IEEE Vehicular Technology Conference (VTC-Fall'11)*, 2011.

[218] Amina Seferagić, Jeroen Famaey, Eli De Poorter, and Jeroen Hoebeke. Survey on Wireless Technology Trade-Offs for the Industrial Internet of Things. *Sensors*, 20(2), 2020.

[219] Khaled Serag, Rohit Bhatia, Akram Faqih, Muslum Ozgur Ozmen, Vireshwar Kumar, Z Berkay Celik, and Dongyan Xu. ZBCAN: A Zero-Byte CAN Defense System. In *Proceedings of the 32nd USENIX Security Symposium (USENIX Sec'23)*, 2023.

[220] Martin Serror, Sacha Hack, Martin Henze, Marko Schuba, and Klaus Wehrle. Challenges and Opportunities in Securing the Industrial Internet of Things. *IEEE Transactions on Industrial Informatics*, 17(5), 2021.

[221] Martin Serror, Eric Wagner, René Glebke, and Klaus Wehrle. QWIN: Facilitating QoS in Wireless Industrial Networks Through Cooperation. In *Proceedings of the IFIP/IEEE Networking Conference (NETWORKING'20)*, 2020.

[222] Faisal Karim Shaikh and Sherali Zeadally. Energy harvesting in wireless sensor networks: A comprehensive review. *Renewable and Sustainable Energy Reviews*, 55, 2016.

[223] Justine Sherry, Chang Lan, Raluca Ada Popa, and Sylvia Ratnasamy. Blindbox: Deep Packet Inspection over Encrypted Traffic. In *Proceedings of the ACM Conference on Special Interest Group on Data Communication (SIGCOMM'15)*, 2015.

[224] Jiwoo Shin, Hyunghoon Kim, Seyoung Lee, Wonsuk Choi, Dong Hoon Lee, and Hyo Jin Jo. RIDAS: Real-time identification of attack sources on controller area networks. In *Proceedings of the 32nd USENIX Security Symposium (USENIX Sec'23)*, 2023.

[225] Simon Sidon. Ein Satz über trigonometrische Polynome und seine Anwendung in der Theorie der Fourier-Reihen. *Mathematische Annalen*, 106(1), 1932.

[226] Marcos A Simplicio Jr, Bruno T De Oliveira, Cintia B Margi, Paulo SLM Barreto, Tereza CMB Carvalho, and Mats Näslund. Survey and comparison of message authentication solutions on wireless sensor networks. *Ad Hoc Networks*, 11(3), 2013.

[227] Emiliano Sisinni, Abusayeed Saifullah, Song Han, Ulf Jennehag, and Mikael Gidlund. Industrial iInternet of Things: Challenges, Opportunities, and Directions. *IEEE Transactions on Industrial Informatics*, 14(11), 2018.

[228] Hyun Min Song, Ha Rang Kim, and Huy Kang Kim. Intrusion Detection System Based on the Analysis of Time Intervals of CAN Messages for In-Vehicle Network. In *Proceedings of the International Conference on Information Networking (ICOIN'16)*, 2016.

[229] Christopher Szilagyi. *Low cost multicast network authentication for embedded control systems.* PhD thesis, Carnegie Mellon University, 2012.

[230] Christopher Szilagyi and Philip Koopman. A Flexible Approach to Embedded Network Multicast Authentication. In *Proceedings of the 2nd Workshop on Embedded Systems Security (WESS'08)*, 2008.

[231] Christopher Szilagyi and Philip Koopman. Flexible Multicast Authentication for Time-Triggered Embedded Control Network Applications. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems & Networks (DSN'09)*, 2009.

[232] Christopher Szilagyi and Philip Koopman. Low Cost Multicast Authentication via Validity Voting in Time-Triggered Embedded Control Networks. In *Proceedings of the 5th Workshop on Embedded Systems Security (WESS'10)*, 2010.

[233] Ted H Szymanski. Supporting Consumer Services in a Deterministic Industrial Internet Core Network. *IEEE Communication Magazine*, 54(6), 2016.

[234] Texas Instruments. SN65HVD25x Turbo CAN Transceivers for Higher Data Rates and Large Networks Including Features for Functional Safety. Technical report, 2015.

[235] A. Richard Thompson, James M. Moran, and George W. Swenson Jr. *Interferometry and Synthesis in Radio Astronomy.* 2017.

[236] Gilles Thonet, Patrick Allard-Jacquin, and Pierre Colle. ZigBee – WiFi Coexistence. *Schneider Electric White Paper and Test Report*, 1, 2008.

[237] Bohdan Trach, Alfred Krohmer, Franz Gregor, Sergei Arnautov, Pramod Bhatotia, and Christof Fetzer. Shieldbox: Secure Middleboxes Using Shielded Execution. In *Proceedings of the Symposium on SDN Research (SOSR'18)*, 2018.

[238] Patrick P Tsang and Sean W Smith. YASIR: A Low-Latency, High-integrity Security Retrofit for Legacy SCADA Systems. In *Proceedings of the IFIP International Information Security Conference (IFIP Sec'08)*, 2008.

[239] Mostafa Uddin, Sarit Mukherjee, Hyunseok Chang, and TV Lakshman. SDN-based service automation for IoT. In *Proceedings of the 25th International Conference on Network Protocols (ICNP'17)*, 2017.

[240] Jo Van Bulck, Jan Tobias Mühlberg, and Frank Piessens. VulCAN: Efficient Component Authentication and Software Isolation for Automotive Control Networks. In *Proceedings of the 33rd Annual Computer Security Applications Conference (ACSAC'17)*, 2017.

[241] Anthony Van Herrewege, Dave Singelee, and Ingrid Verbauwhede. CANAuth - A Simple, Backward Compatible Broadcast Authentication Protocol for CAN bus. In *Proceedings of the ECRYPT Workshop on Lightweight Cryptography*, 2011.

[242] Jonathan Vestin, Andreas Kassler, Sándor Laki, and Gergely Pongrácz. Toward In-Network Event Detection and Filtering for Publish/Subscribe Communication Using Programmable Data Planes. *IEEE Transactions on Network and Service Management*, 18(1), 2020.

[243] Dinh Tung Vo, Xuan Phuong Nguyen, Thai Duong Nguyen, Rahmat Hidayat, Thanh Tung Huynh, and Dinh Tuyen Nguyen. A review on the internet of thing (IoT) technologies in controlling ocean environment. *Energy Sources, Part A: Recovery, Utilization, and Environmental Effects*, 47(1), 2025.

[244] Mehmet C Vuran and Ian F Akyildiz. Cross-layer packet size optimization for wireless terrestrial, underwater, and underground sensor networks. In *Proceedings of the Conference on Computer Communications (INFOCOM'08)*, 2008.

[245] Eric Wagner, Frederik Basels, Jan Bauer, Till Zimmermann, Klaus Wehrle, and Martin Henze. CAIBA: Multicast Source Authentication for CAN Through Reactive Bit Flipping. In *Proceedings of the 2025 IEEE 10th European Symposium on Security and Privacy (EuroS&P'25)*, 2025.

[246] Eric Wagner, Jan Bauer, and Martin 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 (WiSec'22)*, 2022.

[247] Eric Wagner, David Heye, Martin Serror, Ike Kunze, Klaus Wehrle, and Martin Henze. Madtls: Fine-grained Middlebox-aware End-to-end Security for Industrial Communication. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security (AsiaCCS'24)*, 2024.

[248] Eric Wagner, David Heye, Martin Serror, and Klaus Wehrle. Integrating MAC Aggregation over Lossy Channels in DTLS 1.3. In *Proceedings of the 33rd IEEE International Conference on Network Protocols (ICNP'25)*, 2025.

[249] Eric Wagner, Nils Rothaug, Konrad Wolsing, Lennart Bader, Klaus Wehrle, and Martin Henze. Retrofitting Integrity Protection into Unused Header Fields of Legacy Industrial Protocols. In *Proceedings of the 48th IEEE Conference on Local Computer Networks (LCN'23)*, 2023.

[250] Eric Wagner, Martin Serror, Klaus Wehrle, and Martin Henze. BP-MAC: Fast Authentication for Short Messages. In *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec'22)*, 2022.

[251] Eric Wagner, Martin Serror, Klaus Wehrle, and Martin 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'24)*, 2024.

[252] Qian Wang, Yiming Qian, Zhaojun Lu, Yasser Shoukry, and Gang Qu. A Delay based Plug-in-Monitor for Intrusion Detection in Controller Area Network. In *Proceedings of the Asian Hardware Oriented Security and Trust Symposium (AsianHOST'18)*, 2018.

[253] Qiyan Wang and Sanjay Sawhney. VeCure: A Practical Security Framework to Protect the CAN bus of Vehicles. In *Proceedings of the International Conference on the Internet of Things (IOT'14)*, 2014.

[254] Waveshare. Rs485 Can Hat. `https://www.waveshare.com/wiki/RS485_CAN_HAT`. Last Accessed: 12.10.2025.

[255] Yi-Hung Wei, Quan Leng, Song Han, Aloysius K. Mok, Wenlong Zhang, and Masayoshi Tomizuka. RT-WiFi: Real-time high-speed communication protocol for wireless cyber-physical control applications. In *Proceedings of the 34th Real-Time Systems Symposium (RTSS'13)*, 2013.

[256] Haohuang Wen, Qi Alfred Chen, and Zhiqiang Lin. Plug-N-Pwned: Comprehensive Vulnerability Analysis of OBD-II Dongles as A New Over-the-Air Attack Surface in Automotive IoT. In *Proceedings of the 29th USENIX Security Symposium (USENIX Sec'20)*, 2020.

[257] Matthias Wilhelm, Ivan Martinovic, Jens B. Schmitt, and Vincent Lenders. Short Paper: Reactive Jamming in Wireless Networks—How Realistic is the Threat? In *Proceedings of the fourth ACM conference on Wireless Network Security (WiSec'11)*, 2011.

[258] Konrad Wolsing et al. Can Industrial Intrusion Detection Be SIMPLE? In *Proceedings of the European Symposium on Research in Computer Security (ESORICS'22)*, 2022.

[259] Konrad Wolsing, Eric Wagner, Luisa Lux, Klaus Wehrle, and Martin Henze. GeCos Replacing Experts: Generalizable and Comprehensible Industrial Intrusion Detection. In *Proceedings of the 34th USENIX Security Symposium (USENIX Sec'25)*, 2025.

[260] Konrad Wolsing, Eric Wagner, Antoine Saillard, and Martin Henze. IPAL: Breaking Up Silos of Protocol-Dependent and Domain-Specific Industrial Intrusion Detection Systems. In *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses (RAID'22)*, 2022.

[261] Samuel Woo, Hyo Jin Jo, and Dong Hoon Lee. A Practical Wireless Attack on the Connected Car and Security Protocol for In-Vehicle CAN. *IEEE Transactions on Intelligent Transportation Systems*, 16(2), 2015.

[262] Andrew K Wright, John A Kinast, and Joe McCarty. Low-latency Cryptographic Protection for SCADA Communications. In *Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS'04)*, 2004.

[263] Kaiyuan Yang, David Blaauw, and Dennis Sylvester. Hardware Designs for Security in Ultra-Low-Power IoT Systems: An Overview and Survey. *IEEE Micro*, 37(6), 2017.

[264] Sophia Yoo and Xiaoqi Chen. Secure Keyed Hashing on Programmable Switches. In *Proceedings of the ACM SIGCOMM Workshop on Secure Programmable Network Infrastructure (SPIN'21)*, 2021.

[265] Clinton Young, Habeeb Olufowobi, Gedare Bloom, and Joseph Zambreno. Automotive Intrusion Detection Based on Constant CAN Message Frequencies Across Vehicle Driving Modes. In *Proceedings of the ACM Workshop on Automotive Cybersecurity (AutoSec'19)*, 2019.

[266] Sebastian Zander, Grenville Armitage, and Philip Branch. A Survey of Covert Channels and Countermeasures in Computer Network Protocols. *IEEE Communications Surveys & Tutorials*, 9(3), 2007.

[267] Mengqi Zhan, Yang Li, Guangxi Yu, Yan Zhang, Bo Li, and Weiping Wang. GUARDBOX: A high-performance middlebox providing confidentiality and integrity for packets. *IEEE Transactions on Information Forensics and Security*, 2023.

[268] Chenxi Zhang, Xiaodong Lin, Rongxing Lu, and P-H Ho. RAISE: An efficient RSU-aided message authentication scheme in vehicular communication networks. In *Proceedings of the International Conference on Communications (ICC'08)*, 2008.

[269] Collin Zhang, Zachary DeStefano, Arasu Arun, Joseph Bonneau, Paul Grubbs, and Michael Walfish. Zombie: Middleboxes that Don't Snoop. In *IEEE Symposium on Security and Privacy (S&P'23)*, 2023.

[270] Lu Zhou, Kuo-Hui Yeh, Gerhard Hancke, Zhe Liu, and Chunhua Su. Security and Privacy for the Industrial Internet of Things: An overview of approaches to safeguarding endpoints. *IEEE Signal Processing Magazine*, 35(5), 2018.

[271] Ting Zhu, Ziguo Zhong, Tian He, and Zhi-Li Zhang. Exploring Link Correlation for Efficient Flooding in Wireless Sensor Networks. In *Proceedings of the 7th USENIX conference on Networked Systems Design and Implementation (NSDI'10)*, 2010.

List of Acronyms

ACK	Acknowledgement
AD	Associated Data
AEAD	Authenticated Encryption with Associated Data
BitW	Bump-in-the-Wire
CAN	Controller Area Network
CRC	Cyclic Redundancy Check
DoS	Denial of Service
ECDSA	Elliptic Curve Digital Signature Algorithm
ECU	Electronic Control Unit
EOF	end-of-frame
GCM	Galois/Counter Mode
HARQ	Hybrid Automatic Repeat Request
HMAC	Hash-based Message Authentication Code
ICS	Industrial Control System
IDS	Intrusion Detection System
ID	Identifier
IIoT	Industrial IoT
IKE	Internet Key Exchange
IV	Initialization Vector
IoT	Internet of Things
LPWAN	Low-power Wide-area Network
MAC	Message Authentication Code
NRZ	non-return-to-zero
PDU	Protocol Data Unit
PER	Packet Error Rate
ProMAC	Progressive MAC
RBF	Reactive Bit Flipping
SDCC	Software-defined CAN Controller
SIP	Semiconductor Intellectual Property
SOF	start-of-frame

SoC System-on-a-Chip
TCP Transmission Control Protocol
TEE Trusted Execution Environment
TLS Transport Layer Security
UDP User Datagram Protocol

A

Analysis of Selected CAN Protection Mechanisms

A recent survey on CAN authentication schemes [152] revealed weaknesses with many of the proposed CAN authentication schemes but classified others as *secure protocols* (LiBrA-CAN [90], LinAuth [147], LCAP [100], CaCAN [131], and AuthentiCAN [159]). With regard to the eleven drawbacks defined in subsubsection 5.1.2.2, we took a closer look at the latter schemes (*cf.* Subsections A.1 – A.6), as well as some additional authentication schemes not considered by the authors (*i.e.*, Watermarking [162], CANTO [91], ZBCAN [219], CAN-MM [184], LEAP [153], and CAN-TORO [92] in section A.8 – A.12), and re-assess their security in this section. Overall, we find that none of the current schemes offer full protection (as shown in Table 5.1), hence justifying the existence of CAIBA.

A.1 LiBrA-CAN

Description. LiBrA-CAN [90] uses a Mixed MAC approach for integrity protection, *i.e.*, keys are shared among a subset of nodes and each node can only verify a fraction of each authentication tag. These keys are initially distributed by computationally superior master node. To authenticate a frame sent to the bus, the sending ECU first authenticates it to the master node or an helper node. This additional node knows additional keys to compute additional authentication data such that the remaining ECUs can eventually (more than one helper node may be required) verify the integrity of the original message.

Analysis. LiBrA-CAN introduces verification delays and communication overhead, similarly to TESLA-based protocols [188, 189]. This overhead grows linearly with the number of nodes and, for a certain number of nodes, the use of digital signatures

may even become beneficial over using LiBrA-CAN [90]. Additionally, a master node with knowledge of all secret keys exposes a single point of failure, as an attacker that compromises this node gains full control over the network.

Takeaway. LiBrA-CAN's scalability is limited, the authentication requires the sending of additional frames that lead to a verification delay, and the master node offers an attractive target to attackers as it is in possession of all keys.

A.2 LinAuth

Description. In LinAuth [147], each ECU pair shares a secret key. For each CAN ID an ECU transmits or receives, the ECU additionally needs to know the group of other ECUs that are also interested messages with this ID. A sender of a message computes an authentication tag for each interested receiving ECU. The available space is then evenly split among each receiver and fills with the truncated authentication tags as well as the least-significant bits of each ECU-specific counter. Each receiving ECU knows where its authentication data starts and ends within a frame and can thus verify the truncated authentication tag.

Analysis. LinAuth requires significant configuration and storage to track which node is interest in which CAN IDs. Moreover, LinAuth does not scale to realistic network sizes. Reserving 24 bit for authentication data and transmitted 4 bit for each counter, as proposed by the `SecOC Profile 3 (JASPAR)` of AUTOSAR [22], already leaves no space for authentication data with 6 receiving ECU.

Takeaway. LinAuth requires significant upfront configuration and additional storage and does not scale to realistic network sizes regarding the number of ECUs that are interested in a given CAN ID as space in each frame must be reserved for each receiver.

A.3 LCAP

Description. In LCAP [100], CAN frames are authenticated by including a 2-byte "magic number" in the payload. This magic number is an element of a hash chain only known to the sending ECUs of a given communication group, while the receiving ECUs know the previous element of this chain. Thus, upon reception of a frame, receivers can verify that the magic number indeed stems from the alleged sender.

Analysis. In LCAP, all communication groups must be known in advance and the sender in each such group must precompute and store a hash chain of a significant length, amounting to several kB of data per group. If all elements of a chain are consumed, a new chain must be computed and handshake messages must be exchanged on the bus, leading to protocol and communication overhead. Most importantly, LCAP is vulnerable to the hijacking of magic numbers. By decoding the magic number of a payload while immediately jamming the bus afterwards, an attacker

could ensuring the frame is rejected by other ECUs. Before the original frame is retransmitted by the sender, the attacker can now exploit this (still valid) magic number to inject allegedly legitimate messages.

Takeaway. LCAP produces storage and comunication overhead and is vulnerable to frame interceptions.

A.4 CaCAN

Description. Similarly to CAIBA, CaCAN [131] relies on an authenticator node to assist in authenticating CAN frames. This authenticator shares a secret key with each ECUs and has the capability of overwrite in-transmission frames with an error frame, ensuring that the frame is not received by other ECUs. A sending ECU computes a MAC for the message including a counter and transmits the message, the least-significant bits of the counter, and the first byte of the MAC in a payload of a CAN frame. The authenticator verifies this MAC and disrupts the message if the verification fails.

Analysis. In CaCAN no mechanism for resynchronization for the counter after lost frames is devised. Concerning vulnerabilities, CaCAN's authenticator has access to all keys, an attack does not need to compromise it to circumvent CaCAN's protection. It is sufficient to disconnect the authenticator from bus, to disable all authenticity verification without anyone noticing. Afterwards, the attacker can masquerade as any ECU through an attached or compromised ECU as with legacy CAN networks.

Takeaway. Disconnecting CaCAN's authenticator disable all security measures. Additionally, the authenticator may be compromised to gain access to all key and masquerade as other devices. Finally, frame loss leads to desynchronizations of counters that cannot be recovered from.

A.5 MAuth-CAN

Description. MAuth-CAN [113] also relies on an authenticator node to authenticate CAN frames. For each CAN ID, the sending ECU and authenticator establish a session key. With this key, a 32-bit authentication tag is generated and included in each CAN frame, split into the extended identifier and payload fields. The authenticator verifies this tag and broadcasts a report only if the verification fails. Receiving ECUs thus wait for a predefined time after frame detection for this report and only process the message when no report got received. If a report is received, its authenticity if verified by the ECU and the frame it reports on is discarded.

Analysis. The MAuth-CAN authenticator knows all keys and if it is disconnected, no frame verification takes place and no reports are sent. The receiving ECUs can, however, not differentiate if reports are not sent because the frame is authentic or because the authenticator is not operational. Thus, after disconnecting the

authenticator, the attack can masquerade as any ECU through an attached or compromised ECU. Even without attacks, MAuth-CAN introduces a delay for all frames as ECUs wait for a potential report, and resynchronization requires to perform new session key exchange.

Takeaway. Disconnecting the authenticator disable all security measures. Then, or if the authenticator is compromised to gain access to all key and masquerade as other devices. Moreover, all frames are buffered by the receiver and resynchronizations is expensive.

A.6 AuthentiCAN

Description. In AuthentiCAN [159], a nonce list is exchanged between each ECU pair, secured by asymmetric encryption based on a broadcasted public key. The payload of each frame then consists of the encrypted concatenation of a message and the first yet unused nonce from the list. The receiver of a message can decrypt a message and verify that the transmitted nonce matches the expected nonce, which allegedly verifies the authenticity of a message.

Analysis. AuthentiCAN introduces significant overhead w.r.t. to memory and bandwidth consumption to exchange and keep track of the nonce lists and consequently primarily addresses CAN-FD with its higher data rate. Secondly, AuthentiCAN only protects the communication between two ECUs and does not support broadcast communication, *i.e.*, frames intended for multiple receivers.

Takeaway. AuthentiCAN does not support broadcast communication and causes significant memory and communication overhead.

A.7 Watermarking

Description. The idea of watermarking [162] is to overlay a high-frequency signal over ordinary CAN transmissions. These overlayed signals transmit a time-varying watermark, generated by a random number generator that is seeded with a key known to all legitimate devices connected to the bus.

Takeaway. A compromised ECU can transmit messages with legitimate watermarks, enabling the attack to masquerade as any other device.

A.8 CANTO

Description. CANTO [91] provides frame authentication through covert timing channels. It assumes that all CAN IDs are transmitted in a regular pattern that is known in advance by all receivers, which is assisted by an optimal a priori frame scheduling to maximize inter-frame spacing and avoid collisions. At the scheduled

time, a sending ECU computes a MAC of the frame with a shared group key and uses it to derives an additional delay for the frame. The receiving ECUs then verify that this delay from the expected arrival time matches the expectation for the transmitted frame.

Analysis. Because of its assumption that traffic is sent in repeating patterns, the real-world applicability of CANTO to protect in-vehicule communication may be limited due to spontaneous actions, *e.g.*, by humans in the loop. Moreover, even if CAN traffic exhibits the required regularity, the deployment of CANTO requires the modification of each ECU to support the modified scheduling. Finally, CANTO relies on a shared group key, thus an attack that compromises one ECU has access to this key and can masquerade as any other ECU.

Takeaway. CANTO does not protect against a single compromised ECU and reschedules the transmission of frames, ultimately limiting deployability in vehicles as all ECUs must support CANTO.

A.9 ZBCAN

Description. In ZBCAN [219], each ECU shares a pairwise secret with an authenticator node. Transmissions do not occur instantaneously, but instead are executed at specific time slots. Therefore, the time after the last transmission is split into discrete time spans, each consisting of a predefined number of time slots. The sending ECU then computes its time slot based on the CAN ID, the secret shared with the authenticator, and an implicit message counter. For a given CAN ID, only a subset of all time slots are available to divide them into priority classes. The authenticator monitors the network and verifies the time slots of each message based on its ID, and overwrite mis-timed messages with an error frame, such that only verified messages are received by the other ECUs.

Analysis. While ZBCAN does not consume any additional bandwidth, it is also susceptible to several weaknesses. First, the priority of IDs can be inverted: Consider a high priority message that is generated just after its time slot has passed. Then, a lower priority message would take precendence within this time span before the high priority message has another chance to be transmitted, delaying potentially critical messages. Secondly, with the proposed time span length of 64 nominal bit times for each priority class ZBCAN only achieves the equivalent security of a 6-bit authentication tag. Thirdly, the authenticator can be simply disconnected from the bus as all messages (spoofed or not) are accepted by the network without an authenticator. Finally, ZBCAN only authenticates the intention of the sender to transmit, but not the content of the frame. Hence, an attacker may wait for a transmission to overwrite its content (as shown in subsection 5.1.6) and the resulting error frame to compromise the channel.

Takeaway. ZBCAN causes some transmission delays, only protects the sender intention to send but not the content of this transmission with relatively low security levels, and does not protect against disconnecting the authenticator from the bus.

A.10 CAN-MM

Description. With CAN-MM [184], ECUs can be incrementally retrofitted with transmitter and receiver modules. These modules compute a MAC based on a frame's content and a group key and then multiplex the transmission of the CAN frame and the MAC with On-Off keying. A receiving ECU equipped with the receiver module can de-mulitplex the transmission and verify the MAC, while all other ECUs decode ordinary CAN frames.

Takeaway. CAN-MM does not protect against masquerading attacks by compromised ECU because of its reliance on group keys.

A.11 LEAP

Description. In LEAP [153], each pair of ECUs share a secret key which is used to compute a keystream by encrypting the CAN ID of a message. The first eleven bits of this keystream are embedded into the payload at a location determined by next bits of the keystream such that an eavesdropper does not know which payload bits consist of the authentication tag. Finally, the remaining bits of the keystream are used to encrypt the payload. The intended receiver of a message can perform these steps in reverse order to decrypt the payload and verify the correct embedding.

Takeaway. LEAP does not support broadcast communication, and requires relatively high amounts of storage for key material to compute authentication tags embedded into the payload.

A.12 CAN-TORO

Description. CAN-TORO [92] proposes to encrypt CAN IDs with order-preserving encryption to hide the sender from eavesdroppers and to authenticate them without interfering with message prioritization. Each legitimate ECU keeps track of a mapping of IDs to encrypted IDs which is derived from a group key and updated regularly, *e.g.*, once a second. Received CAN frames are discarded if they contain an invalid ID, where the valid IDs change constantly.

Analysis. For CAN-TORO, all ECUs software needs to be modified, as individual ECUs not supporting the protocol interferes with prioritization and may lead to the discarding of valid CAN frames. Moreover, an attack has a short period of time to replay a valid ID before the mapping changes or to outright overwrite the payload of a frame. Finally, by compromising a single ECU, an attacker gains access to the group key and can then masquerade as any other ECU.

Takeaway. CAN-TORO is susceptible to ID reuse by frame interception and does not protect against masquerading attacks by compromised ECUs. Moreover, tracking the mapping of encrypted IDs cost valuable storage.

A.13 Other Approaches

All approaches not further analyzed here (CANAuth [241], Car2X [217], WooAuth [261], VeCure [253], LeiA [197], vatiCAN [183], VulCAN [240], TOUCAN [28], and S2-CAN [190]) have their weaknesses discussed in detail by Lotto *et al.* [152].