Retrofitting Integrity Protection into Unused Header Fields of Legacy Industrial Protocols

Eric Wagner*†, Nils Rothaug†, Konrad Wolsing*†, Lennart Bader*, Klaus Wehrle†*, and Martin Henze§*
*Cyber Analysis & Defense, Fraunhofer FKIE · {firstname.lastname}@fkie.fraunhofer.de
†Communication and Distributed Systems, RWTH Aachen University · {lastname}@comsys.rwth-aachen.de
§Security and Privacy in Industrial Cooperation, RWTH Aachen University · henze@cs.rwth-aachen.de

***Abstract*—Industrial networks become increasingly interconnected, which opens the floodgates for cyberattacks on legacy networks designed without security in mind. Consequently, the vast landscape of legacy industrial communication protocols urgently demands a universal solution to integrate security features retroactively. However, current proposals are hardly adaptable to new scenarios and protocols, even though most industrial protocols share a common theme: Due to their progressive development, previously important legacy features became irrelevant and resulting unused protocol fields now offer a unique opportunity for retrofitting security. Our analysis of three prominent protocols shows that headers offer between 36 and 63 bits of unused space. To take advantage of this space, we designed the REtrofittable ProtEction Library (RePeL), which supports embedding authentication tags into arbitrary combinations of unused header fields. We show that RePeL incurs negligible overhead beyond the cryptographic processing, which can be adapted to hit performance targets or fulfill legal requirements.**

***Index Terms*—industrial control systems, retrofitting, integrity protection, message authentication code**

I. Introduction

Cyberattacks on Industrial Control Systems (ICSs) such as Stuxnet, the attacks on the Ukrainian power grid, or German Steel Mill become significant threats to modern society [4], [7]. Still, industrial communication networks are hardly protected against malicious activities as those networks were originally designed as "air-gapped" systems, *i.e.*, as systems not connected to the outside world, and thus assumed resilient against cyber threats [20]. However, recent events show that even air-gapped networks are prone to cyberattacks [4] and the movement to, *e.g.*, interconnected manufacturing plants or remote control, additionally defies these initial assumptions [32].

Consequently, secure variants of legacy industrial communication protocols, *e.g.*, Modbus running on top of TLS [2], or modern protocols such as OPC UA [1], have recently been standardized. Yet, for established facilities, upgrading to these secure protocols is often impossible due to necessary hardware changes, availability requirements, and the associated cost and risks of breaking a running system [12], [13], [32]. Hence, research focused on retrofitting security measures into legacy protocols, with integrity protection being the most important mechanism for ICS security [2], [5], [8]–[11], [15], [16], [30], [31], [35], [39]. However, these proposals focus on specific protocols or scenarios and hardly discuss deployability. This lack of generalizability leaves many real-world deployments of ICSs with an unclear path toward security due to the high diversity of industrial deployments. Moreover, ICS networks are typically managed by a single operator that thus has the autonomy to make breaking changes to protocols, resulting in many flavors of the already diverse protocol landscapes.

In this paper, we propose an adaptable scheme that evades these limitations by embedding integrity protection in unused header fields of industrial communication protocols. The feasibility of this approach has already been demonstrated, *e.g.*, on traditional Internet protocols to realize covert communication channels [41]. By targeting industrial application layer protocols, we can minimize changes to the network, avoid communication overhead entirely, and interface with legacy devices that require the original network traffic. Additionally, our proposal preserves full protocol conformance if required.

To intercept the current growth of protocol-specific security retrofit solutions, we analyze widespread industrial protocol headers and observe that they offer significant unused space that could carry authentication data. This analysis motivates our design of the *Re*trofittable *Prote*ction *L*ibrary (RePeL) as a modular and adaptable security retrofitting solution for legacy ICS communication. RePeL's design eases effortless changes to how a specific protocol is handled (*e.g.*, to leverage additional fields when they are not used in a given deployment) or the rapid adoption of new protocols. Additionally, RePeL offers various authentication mechanisms depending on the available latency bounds, space constraints, regulations, and requirements for replay protection. Our evaluation shows that RePeL can be efficiently deployed natively or as a bump-in-the-wire solution and introduces hardly any overhead beyond the cryptographic computations.

Contributions. To retrofit integrity protection into insecure legacy ICSs protocols, we make the following contributions:

- We propose a novel security retrofitting scheme based on re-purposing (partially) unused fields of industrial communication protocols, validated by a feasibility analysis of three common industrial protocols (Section III).
- We design the modular and adaptable *Re*trofittable *Prote*ction *L*ibrary (RePeL) that provides customizable and easily integrated security solutions to protect industrial networks against cyberattacks (Section IV).
- We implement, open-source, and evaluate RePeL: We demonstrate its low overhead, which makes it a practical

Author manuscript.

solution to retrofit integrity protection for a variety of currently unsecured industrial networks (Section V).

Availability Statement. Our RePeL implementation is available at https://github.com/fkie-cad/RePeL.

II. Retrofitting Security to Industrial Protocols

Networks in ICSs stand in stark contrast to the modern Internet in many regards. Hence, we give a short overview of the current state of industrial communication (Section II-A) and introduce current efforts to retrofit protection against cyberattacks to widely used protocols (Section II-B). Finally, we present our idea, including respective design goals (Section II-C), that improves upon the current state-of-the-art to achieve flexible and resource constraint-aware retrofittable cybersecurity (Section II-D).

A. Background on Industrial Networks

In contrast to modern IT networks, ICS networks do not necessarily connect to the Internet but are traditionally air-gapped, *i.e.*, isolated networks designed with no external connectivity [22]. Further, they typically exhibit recurring and predictable communication patterns, as their primary purpose is the transmission of monitoring information and occasional control commands [18], [38]. Thus, respective industrial communication protocols reflect these schemes in their design, showing various conceptual similarities. Unfortunately, the combination of industry-tailored, partially proprietary protocols without basic security considerations, and the ongoing digitization of ICS networks render current security assumptions obsolete. Porting serial protocols to IP-based communication – with ModbusTCP as a prominent example – eases cyberattacks against the respective ICS networks, demonstrated by a growing number of recent incidents [4]. Meanwhile, retroactively securing such networks poses a significant challenge for several reasons. Proprietary and specialized devices with lifespans of multiple decades are hard to update and costly to replace [26]. Even when (software) updates are possible, hardware constraints often render the implementation of security mechanisms, *e.g.*, TLS, infeasible [32]. Since the implementation of traditional security mechanisms itself might induce violations of real-time requirements [33] or even unavailability of devices (either by processing overhead [40], or misconfiguration), they are generally not applicable for ICS networks, where – in contrast to traditional networks – (physical) safety and process availability are of utmost importance. Consequently, research focuses on the development of tailored, retrofittable security solutions for these legacy networks that interrupt current operations as little as possible [33] while mitigating the risk of cyberattacks.

B. Related Work on Securing Legacy Industrial Networks

Several security retrofit solutions have been proposed over the years that can be divided into two categories: Extensions to legacy industrial protocols [2], [8], [10], [11], [16], [31] and the design of additional devices (so-called bump-in-the-wire devices) that create protected tunnels between them but forwards legacy communication to existing hardware [5], [9], [15], [30], [35], [39]. 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 those simply encapsulate protocols such as ModbusTCP [2] or EtherNet/IP [8] 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 its security guarantees and restricts which messages can be authenticated [10]. Moreover, the transmission of additional data fields in the Ethernet/IP protocol [11], opportunistically embedding tags into free message space [31] or combining error detection and integrity protection by overwriting the CRC checksum in CAN bus messages [16] have been proposed. Similarly, the IEC 62351-6 [3] standard proposes to embed signatures into reserved fields of the GOOSE protocol.

Most security retrofitting solutions do, however, built on bump-in-the-wire solutions to introduce more intrusive changes to the protocol, while still reducing the overhead compared to a full-grown TLS stack. YASIR [35] 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 [9] or append authentication data and change logs (*e.g.*, after protocol translation) to messages [15]. 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 [5]. MARMAC [30], 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 [42] is a bump-in-the-wire 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 flavors deployed in the real world. Moreover, it is hardly discussed how the proposed retrofittable security schemes would be deployed in practice without interfering with ongoing operations. 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.

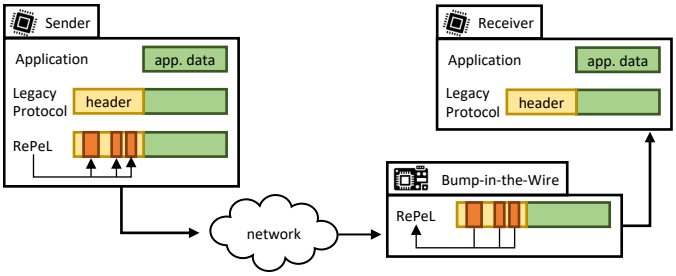


Fig. 1: RePeL can be deployed natively on devices, embedding or authenticating legacy protocol communication as they exit or enter the devices networking stack. Alternatively, RePeL can be deployed as a bump-in-the-wire to segment networks or protect those devices that cannot be updated.

C. Requirements of a Retrofittable Security Solution

Related work mostly 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. To mitigate these limitations of the current state-of-the-art, a *deployable* security retrofit solution must fulfill several 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 be forwarded to the legacy protocol implementation. Specifically, the receiver of a protected packet should be able to recover the originally sent packet in its entirety irrespectively 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 authentication that 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 [27]. Hence, the variety of protocols and constraints in industrial networks demand a flexible and adaptable solution for the retrofitting of data authentication.

Ease of (Incremental) Deployment. Even if security solutions for concrete industrial scenarios exist, limited updateability of devices and resource limitations prevent adoptions 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 bump-in-the-wire 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.

D. The Idea behind RePeL: Recycling Unused Protocol Fields

To achieve these outlined requirements, we design and implement RePeL. RePeL offers a flexible security solution such that a wide variety of network protocols and deployments can be protected by a common solution with minimal configuration. These properties are achieved by integrating authentication data into unused protocol fields of industrial communication protocols, as successfully used to introduce covert channels into traditional Internet communication [41].

The high-level approach of RePeL is illustrated in Figure 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 device (bump-in-the-wire). 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 originally transmitted packet.

E. Threat Model

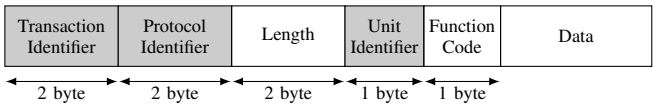
For the design of RePeL, we consider an attack that aims to manipulate messages sent over a communication channel in industrial networks. Our attacker already gained access to the network but has no control over the two entities involved in the targeted communication as defined in the Dolev-Yao threat model [14]. 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-channels attacks are out of scope in this paper as these kinds of attacks affect any security protocol.

III. Analysis of Repurposable Protocol Fields

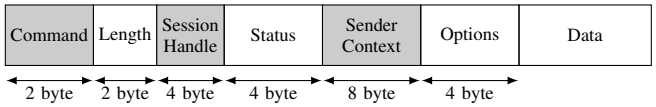
To assess the potential of RePeL, we analyze the feasibility of directly integrating authentication data into unused fields in protocol headers and the subsequent recovery of the original packet without information loss. In particular, we look at three popular ICS protocols, namely EtherNet/IP, PROFINET, and ModbusTCP [19]. Throughout their analysis, we observe common patterns in protocol header designs that RePeL can take advantage of. To this end, Figure 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

Protocol	Header Field	Size	Reusable	Usage Description
ModbusTCP 8 byte header ↓ up to 36 free bits	Transaction Identifier	16 bit	12 bit	As the standard limits the maximum number of concurrently in-flight messages to 16, four bits are sufficient for request/response matching.
	Protocol Identifier	16 bit	16 bit	According to the standard, this field should be set to zero, such that it carries no additional information.
	Unit Identifier	8 bit	0-8 bit	This field is used to identify serial Modbus devices in a ModbusTCP network, which do rarely exist. The field carries no addtional information beyond this identification.
EtherNet/IP 24 byte header ↓ up to 63 free bits	Command	16 bit	7 bit	The first byte is only 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 uniquely identify a communication session. However, even the decently sizes SWAT testbed [17] requires less than 200 session ids over an one hour period.
	Sender Context	64 bit	0-32 bit	Used to match 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.
PROFINET 10 byte header ↓ up to 40 free bits	Service ID	8 bit	3 bit	The first nibble is comprised of only 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 would be used to notify about the embedded authentication data.
	Xid	32 bit	0-16 bit	Used to match 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 two bytes field to extend the length of unicast frames that can be defined to carry authentication data.

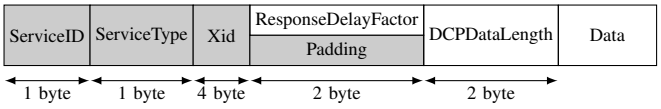
TABLE I: Our analysis of three industrial protocol identifies several fields that can be (partially) used to embed more than 32 bit of authentication data. For each field, we indicate the number of realistically reusable bits and how they could be used.



(a) The ModbusTCP protocol fields.



(b) The EtherNet/IP protocol fields.



(c) The PROFINET protocol fields.

Fig. 2: The three analyzed industrial protocols, namely ModbusTCP, EtherNet/IP, and PROFINET, each exhibit significant potential to embed authentication data (highlighted in gray).

protection in Table I. In the following, we discuss different patterns that we identified among these protocols.

A. 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.

B. 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.

C. 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.

D. 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 protocol standards ask for a static value, no information is lost if the content of such a field is replaced with authentication data. Thus, the recovery of the original packet remains possible.

E. Conclusion

Summarizing our findings, we conclude that unused bits are present in each of the analyzed protocol headers, as detailed in Table I. The most space to embed 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 many 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. In this paper, 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 are negligible to industrial processes running in the background [10]. RePeL adopts to 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.

IV. RePeL: The Retrofittable Protection Library

To take advantage of unused space in industrial protocols headers, we design the *Re*trofittable *P*rote*c*tion *L*ibrary (RePeL).[1] RePeL embeds authentication into protocol 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.

A. High-level Design

RePeL protects legacy traffic between two RePeL endpoints by embedding authentication data into the packets that is later verified and removed. 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.

[1]Code available at: https://github.com/fkie-cad/RePeL

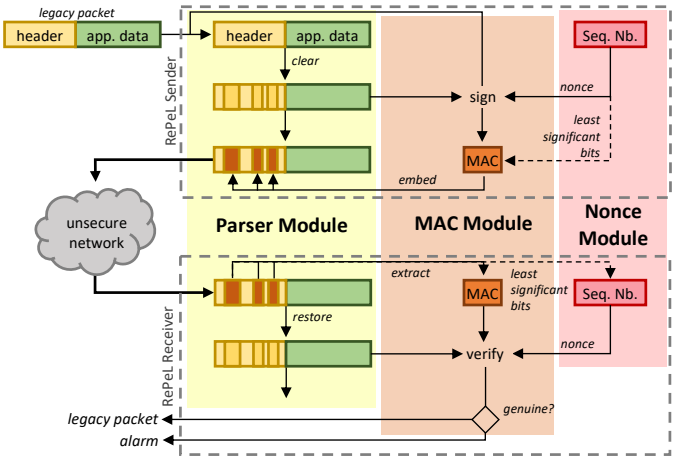


Fig. 3: RePeL is composed of three modules to embed and verify authentication tags. These tags are computed by sender and receiver over a common packet state that each packet is first brought into. If a received packet is genuine, it is forwarded and discarded otherwise.

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 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. 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.

B. 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 managed by a single operator, 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).

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 [28] 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.

C. 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.

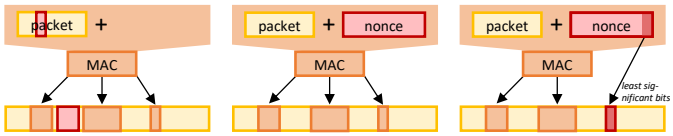
RePeL's parser executes its duties with three passes over a packet for efficient parsing. In the first step, the number of available bits for authentication tags is *extracted*. The second step *clears* the packet of any potential inconsistencies to a form that can also be reproduced by the receiver. This step, for example, includes mapping long message identifiers to their shortened form that 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 adaption to the requirements of vastly different ICS deployments.

D. 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.



(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.

Fig. 4: Depending on the deployment scenario, efficient replay protection is provided with three different mechanisms.

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 is mandated by regulation. However, if stronger security guarantees are required with limited available space, aggregated [21] and progressive MAC schemes [6], [36] may be advantageous. Similarly, latency-critical applications might require fast general schemes such as UMAC [23] or even faster schemes optimized for short messages such as BP-MAC [37].

E. 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 does require space that is often not available. Hence, RePeL offers three methods for replay protection that are shown in Figure 4, which can be selected depending on a scenario's needs.

Figure 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 by itself, 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 which already includes 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 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 [24] and in the proposed DTLS 1.3 standard [29]. 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 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.

V. Performance Analysis

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 [32]. 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 in Section V-A. In Section V-B, we then analyze the latency overhead introduced by RePeL and how it is influenced by packet lengths. Furthermore, in Section V-C, we compare how native security deployments fare against bump-in-the-wire approaches that introduce specific hardware to handle the authentication of legacy traffic.

A. 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 I 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 [27], 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) ▷ 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); ▷ Protocol Identifier
skip(&pkt, 16); ▷ Length
copy(&pkt, 0xff, 8); ▷ Unit Identifier
```

Algorithm 2 then shows how the tag computed over the restored packet by the MAC module is finally 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); ▷ Transaction Identifier
copy(&pkt, &tag, 12);
copy(&pkt, &tag, 16); ▷ Protocol Identifier
skip(&pkt, 16); ▷ Length
copy(&pkt, &tag, 8); ▷ Unit Identifier
```

B. Influence of Packet Length on Delays

To assess RePeL's performance on constrained devices, we first look at the processing time 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

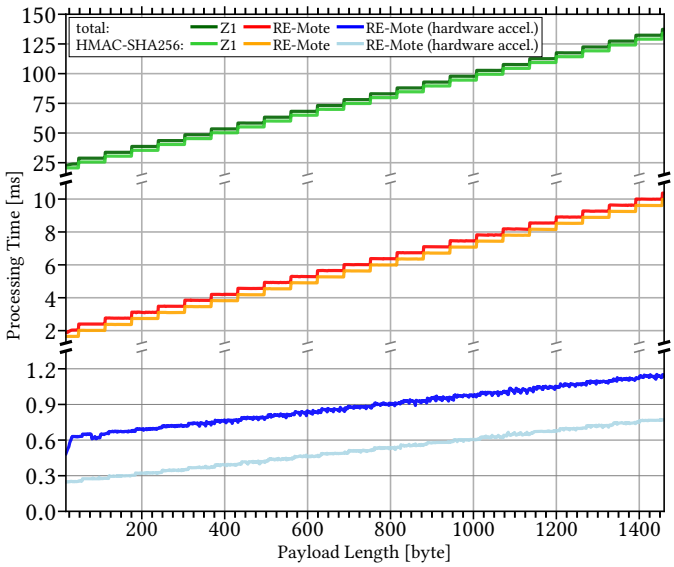


Fig. 5: Native deployments of RePeL on embedded hardware can have vastly different processing times depending on available resources. This processing overhead is dominated by the computations of HMAC-SHA256 digests.

side measurements here. The used authentication scheme is HMAC-SHA256. For each length, we authenticate 1000 packets and repeat this measurement 30 times. Figure 5 includes the 99% confidence intervals, which are barely visible. The measurements are repeated for the Zolertia Z1 and Zolertia RE-Mote. For the RE-Mote, we repeat the measurements also with enabled hardware acceleration for SHA256.

In total, we see significant, but for many applications acceptable [25], 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 lay 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 SHA-256 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 [25].

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

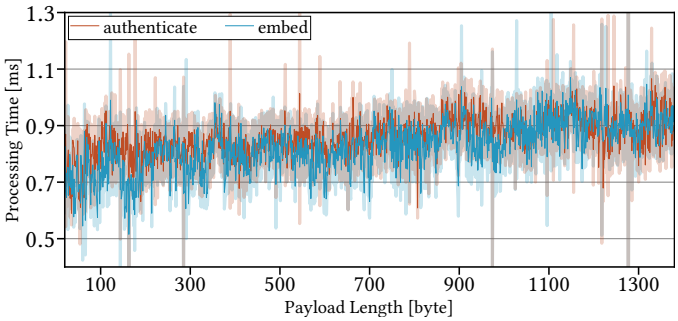


Fig. 6: RePeL deployed on a NanoPi R2S as a bump-in-the-wire reduces processing times at the expense of increased jitter.

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 [23] or BP-MAC [37], if higher performance is required.

C. *RePeL as a bump-in-the-wire deployment*

Many proposals to retrofit security into ICS communication take advantage of a bump-in-the-wire approach to protect the network [5], [9], [15], [30], [35], [39]. 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 bump-in-the-wire deployments. This enables the incremental deployment of true end-to-end security but also helps us to assess the direct performance comparison between the native and bump-in-the-wire security retrofitting solutions.

To this end, we deployed RePeL on a NanoPi R2S to measure the delays 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 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 as evaluated in Section 5, has little influence on the overall delay. Meanwhile, we see significantly higher jitter in these measurements. Hence, bump-in-the-wire deployments offer lower latencies at the expense of increased jitter, which can be problematic in ICSs [34].

VI. Conclusion

The wide variety of legacy industrial protocols 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 bump-in-the-wire solutions ensuring secure communication between RePeL instances.

ACKNOWLEDGMENTS

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2023 Internet of Production – 390621612.

REFERENCES

[1] "OPC Unified Architecture Specification – Part 1: Overview and Concepts," OPC Foundation, Standard, 2017.
[2] "MODBUS/TCP Security Protocol Specification," Modbus Organization, Standard, 2018.
[3] "Power systems management and associated information exchange - Data and communications security - Part 6: Security for IEC 61850," International Electrotechnical Commission, Tech. Rep. IEC 62351-6:2020, 2020.
[4] T. Alladi *et al.*, "Industrial Control Systems: Cyberattack trends and countermeasures," *Comput. Commun.*, vol. 155, 2020.
[5] F. Apolinári *et al.*, "ComSEC: Secure communications for baggage handling systems," in *CPS4CIP*, 2022.
[6] F. Armknecht *et al.*, "ProMACs: Progressive and Resynchronizing MACs for Continuous Efficient Authentication of Message Streams," in *ACM CCS*, 2020.
[7] L. Bader *et al.*, "Comprehensively Analyzing the Impact of Cyberattacks on Power Grids," in *IEEE EuroS&P*, 2023.
[8] B. Batke *et al.*, "Cip security phase 1 secure transport for ethernet/ip," in *ODVA Industry Conference*, 2015.
[9] J. Bauer *et al.*, "CAN't track us: Adaptable privacy for ISOBUS controller area networks," *Comput. Stand. Interfaces*, vol. 66, 2019.
[10] G. Bernieri *et al.*, "TAMBUS: A novel authentication method through covert channels for securing industrial networks," *Comput. Netw.*, vol. 183, 2020.
[11] J. H. Castellanos *et al.*, "Legacy-Compliant Data Authentication for Industrial control System Traffic," in *ACNS*, 2017.
[12] M. Dahlmanns *et al.*, "Easing the Conscience with OPC UA: An Internet-Wide Study on Insecure Deployments," in *IMC*, 2020.
[13] M. Dahlmanns *et al.*, "Missed Opportunities: Measuring the Untapped TLS Support in the Industrial Internet of Things," in *ACM ASIA CCS*, 2022.
[14] D. Dolev *et al.*, "On the Security of Public Key Protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, 1983.
[15] E. Esiner *et al.*, "F-Pro: A Fast and Flexible Provenance-Aware Message Authentication Scheme for Smart Grid," in *SmartGridComm*, 2019.
[22] T. Krause *et al.*, "Cybersecurity in Power Grids: Challenges and Opportunities," *Sensors*, vol. 21, no. 18, 2021.
[16] B. Glas *et al.*, "Signal-based automotive communication security and its interplay with safety requirements," in *ESCAR*, 2012.
[17] J. Goh *et al.*, "A Dataset to Support Research in the Design of Secure Water Treatment Systems," in *CRITIS*, 2016.
[18] J. Hiller *et al.*, "Secure Low Latency Communication for Constrained Industrial IoT Scenarios," in *IEEE LCN*, 2018.
[19] HMS Networks, "Hms networks – fieldbus, industrial ethernet and wireless," 2022.
[20] A. Humayed *et al.*, "Cyber-Physical Systems Security—A Survey," *IEEE Internet Things J.*, vol. 4, no. 6, 2017.
[21] J. Katz *et al.*, "Aggregate message authentication codes," in *Cryptographers' Track at the RSA Conference*. Springer, 2008.
[23] T. Krovetz, "UMAC: Message Authentication Code using Universal Hashing," 2006, RFC 4418.
[24] M. Luk *et al.*, "MiniSec: A Secure Sensor Network Communication Architecture," in *IPSN*, 2007.
[25] M. Luvisotto *et al.*, "Ultra high performance wireless control for critical applications: Challenges and directions," *IEEE Trans. Industr. Inform.*, vol. 13, no. 3, 2016.
[26] S. McLaughlin *et al.*, "The Cybersecurity Landscape in Industrial Control Systems," *Proceedings of the IEEE*, vol. 104, no. 5, 2016.
[27] National Institute of Standards and Technology, "Recommendation for Applications Using Approved Hash Algorithms," U.S. Department of Commerce, Tech. Rep. NIST Special Publication 800-107, 2012.
[28] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," IETF, RFC 8446, 2018.
[29] E. Rescorla *et al.*, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3," IETF, RFC 9147, 2022.
[30] L. Ruhland *et al.*, "Keeping the Baddies Out and the Bridge Calm: Embedded Authentication for Maritime Networks," in *ISNCC*, 2022.
[31] J. Schmandt *et al.*, "Mini-MAC: Raising the bar for vehicular security with a lightweight message authentication protocol," *Veh. Commun.*, vol. 9, 2017.
[32] M. Serror *et al.*, "Challenges and Opportunities in Securing the Industrial Internet of Things," *IEEE Trans. Industr. Inform.*, vol. 17, no. 5, 2021.
[33] E. Sisinni *et al.*, "Industrial Internet of Things: Challenges, Opportunities, and Directions," *IEEE Trans. Industr. Inform.*, vol. 14, no. 11, 2018.
[34] T. H. Szymanski, "Supporting consumer services in a deterministic industrial internet core network," *IEEE Commun. Mag.*, vol. 54, no. 6, 2016.
[35] P. P. Tsang *et al.*, "YASIR: A Low-Latency, High-integrity Security Retrofit for Legacy SCADA Systems," in *IFIP SEC*, 2008.
[36] E. Wagner *et al.*, "Take a Bite of the Reality Sandwich: Revisiting the Security of Progressive Message Authentication Codes," in *ACM WiSec*, 2022.
[37] E. Wagner *et al.*, "BP-MAC: Fast Authentication for Short Messages," in *ACM WiSec*, 2022.
[38] K. Wolsing *et al.*, "IPAL: Breaking up Silos of Protocol-dependent and Domain-specific Industrial Intrusion Detection Systems," in *RAID*, 2022.
[39] A. K. Wright *et al.*, "Low-latency Cryptographic Protection for SCADA Communications," in *ACNS*, 2004.
[40] Z. Yang *et al.*, "Opportunities and Challenges in Securing Critical Infrastructures Through Cryptography," *IEEE Secur. Priv.*, vol. 19, no. 5, 2021.
[41] S. Zander *et al.*, "A Survey of Covert Channels and Countermeasures in Computer Network Protocols," *IEEE Commun. Surv. Tutor.*, vol. 9, no. 3, 2007.
[42] M. Zhan *et al.*, "GUARDBOX: A high-performance middlebox providing confidentiality and integrity for packets," *IEEE Transactions on Information Forensics and Security*, 2023.