A Security Gateway for Power Distribution Systems in Open Networks

Eduardo Andrade[a,*], Jorge Granjal[a], João P. Vilela[a,b], Carlos Arantes[c]

[a] *CISUC, Dep. of Informatics Engineering, University of Coimbra, Portugal*
[b] *CRACS/INESCTEC, Dep. of Computer Science, Faculty of Sciences, University of Porto, Portugal*
[c] *Protection, Automation and Control Division, Efacec, Porto, Portugal*

Abstract

Power Distribution Systems usually rely on closed and fixed communication networks due to the strict requirements they must comply with. With the appearance of new communication technologies that can contribute to the assurance of those requirements (for example, 5G), open networks can be used for such systems, decreasing the overall cost of maintaining and upgrading the communication network. Although, shifting from closed communication environments to networks integrated with the Internet using 5G communication environments can expose these systems to severe threats, since they were developed to operate under closed networks not addressing security by default. This paper analyses the security requirements for Power Distribution Systems operating on open networks, identifying the gap between such systems and the existing security mechanisms. From this analysis, we present a solution based on low cost off-the-shelf hardware, composed by a security library and a bridging device, intended to act as a security gateway for Intelligent Electronic Devices (IEDs) in Power Distribution Systems. We also evaluate the functionality of our security gateway, and analyse its impact on the stringent performance requirements of such systems.

*Corresponding author
Email addresses: eandrade@student.dei.uc.pt (Eduardo Andrade), jgranjal@dei.uc.pt (Jorge Granjal), jvilela@fc.up.pt (João P. Vilela), carlos.arantes@efacec.com (Carlos Arantes)

Keywords: Critical Systems, R-GOOSE Protocol, Power Distribution Systems, Security, IEC 61850

1. Introduction

Power Distribution Systems are responsible to manage the distribution of electrical power over the power grid. These systems are composed by a large number of components, namely Intelligent Electronic Devices (IEDs), substations and reclosers [1]. While it aims to manage the power distribution, the system must be reliable and resilient to anomalous situations, such as unexpected temporary or permanent electrical faults. Unexpected temporary faults can be caused, for example, by a lightning that hits the power line causing an electric arc. Permanent electrical faults can be caused, for example, by a fall of a tree on the power line, breaking the cables.

Nowadays these systems rely on well tested, fixed, closed and cable-based networks [2]. The usage of these networks are explained by the strict performance requirements that these systems must comply with, while at the same time they need to protect the communications from unknown threats (natural or intentional). These systems deal with critical communications, meaning that they must ensure the reliability and resilience of such communication system, and at the same time operate under ultra-low latency requirements. However, these type of networks present limitations when it is necessary to upgrade or add new components to the systems. As an example, deploying a new component on the network may imply the deployment of new network lines, increasing the overall cost of the operation. This makes it very tempting to shift from wired communications to wireless.

Fortunately, there are emerging new communication technologies that may be applied in critical systems. An example of such technology is 5G. The 5G is already being deployed, with the goal of replacing the 4th Generation Long-Term Evolution (4G/LTE). The technological developments provided by 5G will improve the overall mobile communications, in terms of bandwidth, latency and

scalability. This improvements will be able to ensure ultra-low latency and large bandwidth, supporting the performance requirements of such applications and allowing the migration of critical applications to open communications environments. In the particular scenario of Power Distribution Systems, such migration could be particularly interesting given the high number of devices connected to the network, as well as the constant expansion of such network.

1.1. Paper Motivation

This paper addresses the security issues that arise by shifting Power Distribution Systems from closed networks to open networks. 5G communication technologies are becoming available to ensure ultra-low latency communications, therefore favoring the move from closed to open networks and reducing the overall cost of managing such systems. Therefore, this creates the necessity of analysing what are the issues of moving from closed to open networks and how to address them. There are already standards and protocols that provide guidelines on and target these problems, although as we will demonstrate in this work, the proposed solutions may not be able to operate on the old and legacy devices that are placed on these systems, and at the same time comply with the performance requirements. Thus, our motivation is to provide a complete solution that allow to move all devices (new, powerful, legacy and low capability) from closed to open networks.

1.2. Paper Contributions

This paper contributes with the design, implementation and validation of a low cost security gateway to ensure the security requirements while complying with the performance requirements of Power Distribution Systems. This was supported by a security analysis done on shifting such systems from closed networks to open networks.

More precisely, we present the design and implementation of two central components of the security gateway: a security library and a bridging device.

This security library implements a set of security mechanisms that can be integrated directly in IEDs in standalone mode, or through a bridging device to provide security to networks powered by legacy or less capable devices. The security mechanisms implemented follow the guidelines present in several standards such as IEC 61850 [5] and IEC 62351 [6], for data communications within Power Distribution Systems.

We also present the evaluation performed on each component to validate its usability in these systems, evaluating the functionality of the security mechanisms and the performance penalties that our solution will add to the system.

1.3. Paper Organization

This paper is organized as follows. Section 2 presents the background knowledge of Power Distributions Systems and protocols employed. Section 3 presents a a literature review on works that focus the security of Power Distribution Systems, namely in the communication protocols used. Section 4 presents an analysis of security and performance requirements for Power Distribution Systems. Section 5 presents the proposed solution, containing the design and implementation of each component, while Section 6 evaluates the developed components from a performance and security perspective. Finally, Section 7 presents the conclusions and discusses future work.

2. Background Knowledge

Power Distribution Systems are composed by several components, namely IEDs, substations and reclosers [1]. All of these components are traditionally connected over dedicated networks, either inside the same substation (connecting local IEDs) or interconnecting the substation's networks [2].

Managing this kind of systems is very complex, due to the large number of nodes and tendency to keep growing in number of nodes connected to the network. Some techniques used are based on self-healing [18], where the grid has the ability of identifying where the failure occurred and deviate the energy

flow to a redundant line, covering the previous node. An actual challenge is related to the way this self-healing capability is achieved, as some techniques, more precisely the distributed self-healing, needs that various endpoint nodes have the ability to communicate with each other, which is impossible with the technologies used on the field today [19].

These networks usually support standardized network protocol stack architectures, composed by protocols such as Ethernet, IP, TCP/UDP and application layer protocols. At the upper level, the protocols are designed to ensure the high availability, resilience and reliability that these systems require. Some examples of such protocols are GOOSE [5], R-GOOSE [20] and DNP3 [26], all used to transmit messages on power distribution systems. The support of such well-established industry solutions to communications in critical environments provide research challenges related with, among others, security, as we address in this article.

The devices used in these systems are mainly IEDs [36]. These are computer-based controllers of power system equipments, and they operate by collecting data from sensors and issuing commands to other system components, as well as raising events, such as signalling a failure. These components, are of major importance as they allow the grid to be managed and to be supervised.

In this work, we focused on analysing power distribution systems using the protocol R-GOOSE, mainly focusing on shifting from closed and dedicated to open networks, analysing the new threats to these systems and proposing solutions that make such shifting a viable solution. R-GOOSE was chosen because it is representative of a class of critical applications (in particular, power distribution systems) which pose challenges in terms of security, considering the goal of adopting open communications environments [23]. This protocol will be detailed in the next subsection. Following R-GOOSE description, we present an analysis of the most important standards that regulate these systems, more precisely in terms of cyber security to specify the security and performance requirements that our solution must comply with.

2.1. R-GOOSE

An interesting step to the management of power distribution systems is to move from closed and proprietary communication environment to open and wireless networks, allowing an easier integration and deployment of new devices on the network [2] [3]. With this in mind, shifting to IP-based instead of Ethernet based protocols (as GOOSE) is becoming prevalent in the area, motivating several research efforts [2] [3]. For that, the GOOSE protocol was adapted to run over IP-based networks, and consequently Routable-GOOSE (R-GOOSE) Protocol was designed.

R-GOOSE, that is defined on IEC 61850, is an extension of the GOOSE protocol. GOOSE provides fast and reliable mechanisms to maintain intercommunication of substations, by means of multicast or broadcast over Ethernet [4]. As an extension of GOOSE, R-GOOSE provides the same main functions but over IP-based networks.

The main objective of both GOOSE and R-GOOSE is to deliver event data to other nodes on a fast and reliable way. In essence, R-GOOSE messages are the payload of a GOOSE message "wrapped" inside a UDP packet, where GOOSE payloads are data sets of grouped data, as for example in a *(status,value)* format. These messages are event driven, meaning the content or data inside each message are related to or have the objective to generate an event.

As illustrated in Figure 1, R-GOOSE messages can be analysed from the transport and application profile perspectives. From the transport profile perspective, R-GOOSE messages are UDP datagrams, encapsulated inside IP packets. For the application profile, R-GOOSE is composed by several header fields. Among such fields, the most relevant for the security analysis are the *Security Information* fields, and the *Signature* fields. The *Security Information* fields contains data relative to the key used by the security algorithms, namely the *TimeOfCurrentKey*, *TimeToNextKey*, *Security Algorithms* and *KeyID*. The *TimeOfCurrentKey* is a 4-bytes long field representing the time in seconds since the "epoch". The field *TimeToNextKey* is 2-byte long field containing the number of minutes until a new key is used. The first byte of the *Security Algorithms*

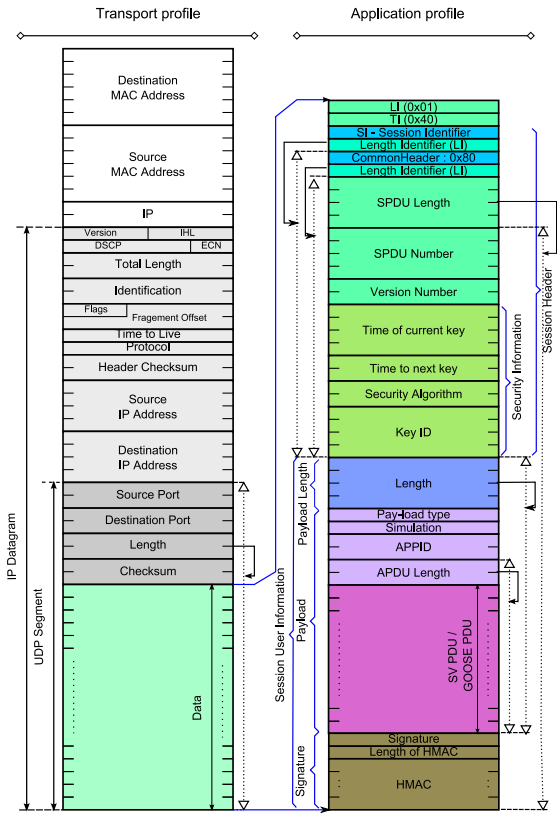

Figure 1: R-GOOSE PDU [8].

field contains the encryption algorithm used, while the second byte contains the algorithm used to generate the MAC. Finally, the *KeyID* field contains the information that identifies the key used, and is set by the Key Management System. Other block of relevant fields are the *Signature* fields. These appear at the end of the R-GOOSE payload and contain the authentication tag for that packet.

2.2. Standards

There are several standards related to the system's design as well as for protocol stacks to be implemented, as for example IEC 61850, that describes a protocol stack to be used for communications on power distribution systems,

with R-GOOSE being one of them. Related to security, there exists one major standard, IEC 62351, that specifies several mechanisms that should be added on top of other standards, as for example IEC 61850, and protocols, such as R-GOOSE. On the next subsections we focus on IEC 61850 and IEC 62351. From the first standard we are going to focus on the performance requirements that these systems must comply with, more precisely in terms of latency. Regarding IEC 62351, we will only focus on parts that are related with the protocols that we used in this work, such as R-GOOSE.

2.2.1. IEC 61850

The analysis of the performance requirements of this system plays an important role on this work, as adding extra security mechanisms will certainly add delay to the communications, being necessary to assess the time window that exists. These requirements are established on the fifth part of IEC 61850 standard, addressing specially the performance requirements for R-GOOSE messages.

The standard specifies maximum message transfer time, given the type of message and the application. The transfer time is for the complete transmission of a message, including the time that devices take to handle it at both ends, meaning it starts being measured when the sender puts some data on the transmission stack, until the moment the receiver retrieves it from its transmission stack.

Given this, the transfer time includes the processing time in both physical devices (t_a and t_c) and the transmission time over the transmission system (t_b), as illustrated in Figure 2.

There is not a common transfer time requirement for transmitting all messages, as different applications and functions inside the system might have different requirements. Messages are divided in types, where they are grouped by similar performance needs. Some message types are also divided in performance classes, as, depending on the context of the event, it may require different timings. On Table 1 are described the transfer time requirements for each message type and performance class.

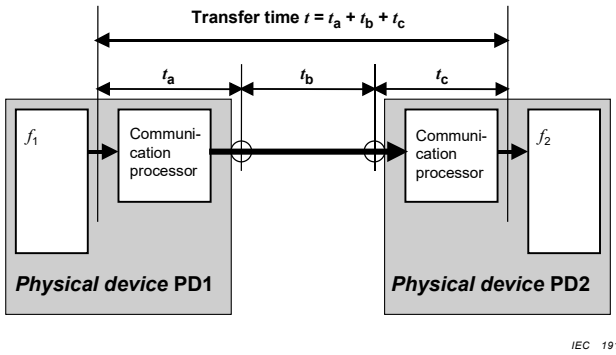

Figure 2: Computation of total transfer time for a message between physical devices (IEC 61850) [5].

Table 1: Table of Maximum Transfer time for each Message Type and Performance Class [5].

Type	P Class	Max. Transfer time	Application
1A	P1	10ms	Fast M. - Trip
	P2/3	3ms	
1B	P1	100ms	Fast M. - Others
	P2/3	20ms	
2	-	100ms	Medium S. M.
3	-	500ms	Low S. M.
4	P1	10ms	Raw Data M.
	P2/3	3ms	
5	-	$\geq$1000ms	File Transfer
6	-	-	Time Sync.
7	-	-	Control Commands

185 We should note that from the standard IEC 61850, the message types that are addressed by R-GOOSE are Type 1A and 1B. In the next subsection we will provide an overview of the relevant parts for this work of IEC 62351 standard, focusing on the security recommendations for GOOSE and R-GOOSE (despite the fact we used R-GOOSE in this work, we will also present the relevant part

from GOOSE due to its similarities).

2.2.2. IEC 62351

This standard was developed to address specifically security issues on systems and protocols used on power distribution environments. The standard addresses security related to the data exchanged among several components of the system, trying to achieve security properties such as confidentiality, integrity, non-repudiation and availability. A relevant aspect of the standard is that it provides specific technical details on the security mechanisms recommended. Inside the scope of this standard is R-GOOSE and IEC 61850. In this case, there are usually strict requirements related to the performance of the system, which makes it difficult to implement good security mechanisms.

The IEC 62351 recommendations for GOOSE are included on part 6. This section presents the recommendations for Layer 2 and peer-to-peer communications. There are two main requirements defined as mandatory: data integrity and data authenticity, while data confidentiality is left as optional. This choice is due to the existence of low capability devices that, when applying confidentiality mechanisms, could compromise the strict latency requirements. To achieve data integrity and authenticity it is recommended to use digital signatures and hash functions with GOOSE messages. The main proposed scheme is to use digital signatures based on RSA.

For R-GOOSE, on older versions of IEC 62351, it was also recommended to add an RSA-based signature to the PDU of the message, ensuring data integrity to those frames. In addition, the PDU could also be encrypted, however in any case the performance requirements should never be compromised. More recently, IEC 62351-6:2020 draft [21] includes the possibility of using hash-based authentication, as HMAC (Hash Message Authentication Code), GMAC (Galois Message Authentication Code), using several hashing algorithms, and AES-GCM (Advanced Encryption Standard - Galois/Counter Mode).

3. Related Work

In this section we will present a literature review on GOOSE and R-GOOSE, focusing on works that analysed its security and the recommended security mechanisms for each protocol. Firstly we will present works that analyse attacks on GOOSE, followed by works that analysed the recommend security mechanisms, mainly in terms of performance. Then, we will present works that focused on R-GOOSE. Finally, we will present works that proposed solutions to performance related problems with security mechanisms in GOOSE and R-GOOSE.

To support the need for authentication, integrity and confidentiality for GOOSE and R-GOOSE, there are several works showing attacks that can compromise the system that could be prevented if security mechanisms were in place. Kush et al. [33] looked at how GOOSE receivers (subscribers) processed the messages, identifying a severe vulnerability related with the usage of the *StNum* field. The vulnerability consisted on the message validation. The authors modified the StNum field to exploit such vulnerability, and were able to create three types of flooding attacks. During this attacks, they were also able to invalidate true-valid messages. In complement Hoyos et al. [34], proved that was also possible to perform spoofing attacks exploiting the StNum properties.

Regarding the recommendations from the standards there are there several important works that we should note. Oberweier et al., in [9], point out some restrictions on the standard. In this work, they analysed the standard recommendations on GOOSE, namely the RSA-based authentication. The authors state that this authentication scheme is not suitable, in terms of performance, for applications with ultra-low latency requirements, as is the case of power distribution systems. Ishchenko et al., in [32], did a similar analysis on RSA-based schemes and they confirmed that RSA-based digital signatures were not a feasible solution for GOOSE (and R-GOOSE) messages, as its computation would take some milliseconds.

Farooq et al. in [10], analysed the algorithms and schemes recommended

by IEC 62351-6 to understand if it is a feasible solution to ensure authenticity and integrity on GOOSE messages. In particular, the authors analysed the performance of RSASSA-PSS, RSA e RSASSA-PKCS1-v1_5 based schemes, with key lengths of 1024 and 2048 bits. Using a low capability device, the authors realized that RSASSA-PKCS1-v1 was the best performing algorithm. However, securing a GOOSE message would still take 0.9 ms with 1024 bits and 3.56 ms with 2048 bits, concluding that this is not a feasible solution. The same conclusion was also demonstrated in [11]. In [9] and [10], the authors propose the usage of HMAC, both in hardware or software implementations, given that its computational overhead is significantly smaller and it complies with the time restrictions.

Following the work done on [10], Farooq et al. evaluated the performance of MAC based algorithms for GOOSE messages on [27]. In this work, the authors developed a library containing a set of cryptographic algorithms to ensure authentication and integrity on GOOSE messages, namely HMAC-SHA256 (truncated to 80, 128, 256 bits), AES-GMAC-64 e AES-GMAC-128. The performance of such algorithms was analysed using an Intel®Celeron(R) processor with 4 GB RAM, being a relatively old and slow system. From this work, the authors concluded that MAC based algorithms are suitable for ultra-low latency requirements as in GOOSE.

Later, Farooq et al. on [28], focused on the lack of encryption for GOOSE messages. They proposed three models for usage of encryption with authentication: Encrypt-then-MAC (EtM), Encrypt-and-MAC (E&M) and MAC-then-Encrypt (MtE). They use HMAC-SHA256 to support authentication and AES-128 for symmetric encryption. From their work they concluded it was possible to use authentication and encryption for GOOSE messages using such algorithms.

Continuing the same work, Farooq et al. analysed how digital signatures based on RSA and ECDSA (Elliptic Curves) could be used to protect GOOSE messages against replay and masquerade attacks on [29]. After analysing several attack scenarios, they analysed their solution using RSA based algorithms with several key lengths (1024, 2048, 3072 bits) and ECDSA with different curves,

measuring the time to sign and validate messages. The data collected showed that DS based methods are not the best performing, and even the ECDSA based algorithms can compromise the performance requirements.

On [30], Farooq et al. developed a testbed for R-GOOSE and R-SV, applied the previously developed mechanisms to R-GOOSE and analysed its performance on a low performance (legacy) computer. They concluded that it was possible to apply such mechanisms to R-GOOSE messages with low capability devices. In terms of R-GOOSE performance analysis, our work complements the previous by analysing the performance of such algorithms in real IED devices with industrial specifications, as described in Section 6.1. Therefore, we validated our proposal towards its implementation in real application scenarios. Also, we considered and included, in our library, a set of complementary authentication mechanisms, namely those specifically designed to run on low-capability devices such as BLAKE2b.

Finally, Rodríguez et al. in [31] proposed a Security Gateway for GOOSE communications. This was an hardware-based implementation, using an FPGA from Xilinx Zynq-7020 family. Our work differs from this one in terms of the implementation and target protocol. As mentioned, this work was targeting GOOSE protocol, and even if similar, GOOSE and R-GOOSE operate in different protocol layers. Moreover, the inside and formatting of each protocol packet is different, requiring different implementations. Given this, our work and [31] can be applied in different scenarios. As mentioned by Apostolov in [37], a GOOSE based solution was originally designed to transmit messages inside substations, while R-GOOSE based solutions target communications between substations, although, it can also be used for inside substation communications. Also, in this work the authors developed an hardware-based solution, while our solution is software-based. As we will demonstrate later in this document, our solution can be applied directly on real IEDs, or can be used separately as a Security Gateway for low capability IEDs.

As these works demonstrated, currently operating R-GOOSE implementations do not include security by default, as such features could compromise the

strict performance requirements, specifically on old and legacy devices. This challenge motivates our proposal, whereby we develop a Security Library that implements all recommended security mechanisms for R-GOOSE, and a Security Gateway to allow the usage of such security mechanisms with legacy and low-capability devices.

4. Security and Performance Requirements Analysis

In this section, we will present and analyse the security and performance requirements that Power Distribution Systems, focusing on communications based on R-GOOSE, must or should comply with. We supported this analysis on the related work already discussed, as well as on the relevant standards and recommendations, such as IEC 61850 and IEC 62351.

As the Power Distribution Systems are critical systems, where a fault could create huge damage, either at the infrastructure itself, as well as endangering human lives, it is extremely important to ensure that the system has high availability. This leads us to the first requirement of System Availability. Furthermore, from the analysis done on IEC 62351, we recognise that message authentication and message integrity is also a must. Naturally, as we may face unauthorized access we must ensure that only valid messages, that were sent by authorized devices will be accepted, as well as only messages that were not tampered are accepted. This reasoning is supported by several works presented previously, as for example in [33] and [34]. In the same standard, we can see that confidentiality should also be considered, specially in R-GOOSE that operates over open networks, although, this should not compromise the timeliness and low latency/response time of the system. As shown in [35], capturing IEC 61850 traffic will give the attacker the knowledge to be able to craft valid packets with malicious content. Given this, we can also set Message Integrity, Authentication and Confidentiality as requirements.

From the standard IEC 61850 we can specify the exact values that this system has to comply with in terms of latency. In this standard it is specified

that R-GOOSE will deal with the messages of Type 1A and 1B. The values, as shown on the Table 1, are between 3ms and 100ms, meaning that ultra-low latency is required on the communications. This requirement is of particular importance for testing and validation of our solution, as performance can not be decreased to a latency higher than 3ms at the expense of applying security. On Table 2 we summarize the requirements with a brief description.

Table 2: Power Distribution System requirements and their description. Requirements based on references [5] [6]

Requirement	**Description**
System Availability	The system must be available any time it is required, even if a given communication line is down
Message authenticity	Must be possible to identify and confirm who is the message sender
Message integrity	Must be possible to verify if the message received did not suffered any change since it was sent (either by natural or unnatural causes)
Message confidentiality	Must be impossible to an unauthorized user understand the contents of the message, the messages should be encrypted
Message timeliness/very low latency	The messages should be delivered in the minimum time possible, never exceeding the maximum Transfer Time of 3ms/10ms (depending on the scenario)

5. Proposed Solution and Implementation

Our goal is to develop a product to provide security mechanisms capable of ensuring security requirements for critical applications like Power Distribution Systems. From Section 4, we established the need for security mechanisms to ensure Message Integrity, Message Authentication and Message Confidentiality.

It is also crucial that these security mechanisms do not compromise the strict performance requirements of these systems.

Taking these facts into account, the first component we propose is a Security Gateway that will implement a set of security mechanisms (including the algorithms recommended by IEC 62351). As we learn from several related works (and from the experimental analysis presented later using real IEDs), some of the currently being used IEDs may not be able to apply the such security mechanisms and therefore there is the need to find a solution for such devices. With this in mind, we propose a component that is Security Gateway. This security gateway is a bridging device based on COTS hardware that will apply the security mechanisms present on our Security Library. Summarizing, we propose a security framework composed by two main components: a Security Library and a Bridging Device, as we proceed to discuss.

5.1. Security Library

Security Library [12] is a library written in C and using OpenSSL 1.1.1 Library, providing a collection of functions ready-to-use by other applications, to ensure security properties such as integrity, authentication and confidentiality to packets. We can divide the library in two parts: cryptographic algorithms and protocol related functions. The first part contains a set of cryptographic functions, as for example HMAC-SHA256 or AES256-GCM. We used OpenSSL to get well-tested implementations of these algorithms. In the second part of the library we implemented the functions responsible to deal with specific aspects of each protocol as, for example, encrypting the packet payload and changing all the other mutable fields of the protocol, using the cryptographic functions included in the first part.

We developed this library to provide three security properties: Message Integrity, Message Authentication and Message Confidentiality. We selected the algorithms that were recommended by the standard IEC 62351, that provides guidelines for security on R-GOOSE. In this standard is recommended the usage of Hash Message Authentication Code (HMAC) or Galois Message Authentica-

tion Code (GMAC) based authentication. We also selected two other variations of HMAC-based authentication that were not included in IEC 62351, HMAC-BLAKE2b and HMAC-BLAKE2s, with the motivation that these are two algorithms designed to operate in low capability devices. For encryption, we used the algorithms recommended in the same standard, namely AES based encryption with key sizes of 256 and 128 bits. In more detail, for the cryptographic functions our library includes the following features presented on Table 3.

Table 3: Security algorithms for authentication, encryption and decryption included on the security library.

HMAC Generation functions	HMAC-SHA256-80
	HMAC-SHA256-128
	HMAC-SHA256-256
	HMAC-BLAKE2b-80
	HMAC-BLAKE2s-80
GMAC Generation functions	GMAC-AES256-64
	GMAC-AES256-128
	GMAC-AES128-64
	GMAC-AES128-128
AES-GCM encryption and decryption functions	AES128-GCM
	AES256-GCM

These functions are responsible for performing the cryptographic operations on a given data, producing an HMAC or a GMAC as output, or encrypting and decrypting data.

As for the protocol integration functions, in this case, R-GOOSE, we developed the following features:

- R-GOOSE Authentication
    1. Insert and Validate HMAC
    2. Insert and Validate GMAC

- R-GOOSE Encryption

    1. Encrypt R-GOOSE Payload
    2. Decrypt R-GOOSE Payload

These functions resort to the cryptographic functions previously mentioned to provide the security properties to a given protocol (in this case R-GOOSE), dealing with the protocol specific aspects, as for example the mutable fields. Another important goal of the security library is to provide support security for R-GOOSE communications using both legacy IED devices and newer platform (namely RaspberryPi devices), as we will address later in this article.

5.2. Bridging Device

The Bridging Device is the framework component placed between the system component we want to "protect", and the network it is connected to. The device captures the packets sent from the protected device, analyse them and, if necessary, applies the security methods to ensure the set of security requirements. With this goal the bridging device resorts to the Security Library already described to apply the security mechanisms. In terms of hardware, our bridging device is a Raspberry Pi 4B, running the Raspbian Linux variant and equipped with an USB-to-Ethernet adapter, to provide an extra Ethernet port. More precisely, the technical specifications of the Raspberry Pi are presented in the Table 4. The goal with this type of device is to demonstrate that it is possible to use off-the-shelf hardware to improve the security on systems composed by specialized equipments such as IEDs, that are not usually capable of performing such tasks.

This component will allow legacy and lower capacity devices to be protected with the necessary security mechanisms, thus enabling protection for devices that are not capable of performing such tasks. The approach taken was to create a Linux Bridge handled by the kernel. That bridge captures packets on the incoming interface, moves such packets from the kernel space to a user space application. This application modifies the packets and re-sends them again to

Table 4: Raspberry Pi 4B model Technical Specifications.

CPU	Quad core Cortex-A72 (ARM x64) 1.5GHz
RAM	4GB
Ethernet	Gigabit Ethernet
External Ethernet	USB2.0 to Fast Ethernet
Operating System	Raspbian GNU/Linux 10 (Buster)

the kernel, that in turn bridges the secured packet to the outgoing interface. Packets are moved to user space because our security library (Section 5.1) uses the OpenSSL library, which cannot be used in kernel space.

The Linux bridge handled by the kernel was achieved using the *brctl-utils* [24] tool that connects two physical Ethernet interfaces creating a new logical interface to the bridge, being available to use on the Raspbian OS.

After setting up the bridge, it is necessary to move the packets from kernel space to user space. We achieved that using IPTables alongside with another Netfilter module. On IPTables, we used its feature that allows a custom packet filtering, performed in userspace, using NFQUEUE. Using this, a user-space application can set a verdict on a given packet and even modify or craft a new packet, and then re-inject it into the IPTables chain.

Finally, the user space application will be listening for packets moved by IPTables to the NFQUEUE. Netfilter provides a library to manage such queues, being handled the same way as sockets. When a packet is received, we call a callback function to process each packet. This function can be customized and designed to fit any protocol. In our case, we created a set of configurations to process each R-GOOSE packet, depending on the interface the packet arrived and a set of pre-configurations as, for example, the authentication and encryption algorithm to be applied on unsecured packets. When a valid R-GOOSE packet is identified and it is necessary to apply a security mechanism (when a unsecured packet arrives), the developed security library is used and the packet is modified or validated/invalidated.

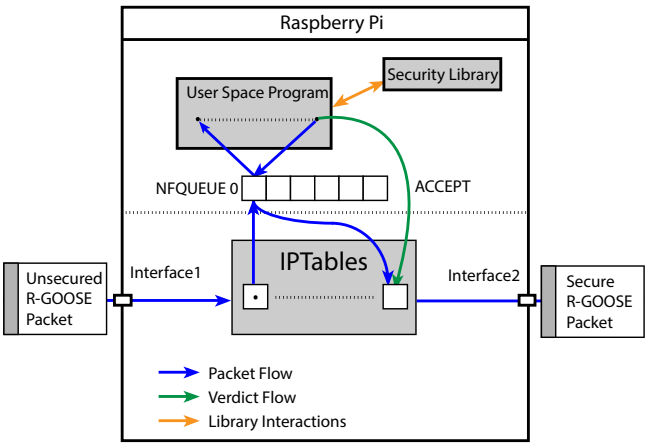

Figure 3: Bridge Diagram - Interaction between IPTables, Netfilter NFQUEUEs and user space program.

Figure 3 illustrates our bridging solution, showing in detail the interactions between IPTables, NFQUEUE, our user space program and the developed security library. The blue arrows represent the path that an arriving packet will transverse, the green arrow represents the verdict issued by our program to IPTables, while the orange arrow represents the interactions between the user space program and the security library. Figure 3 illustrates a situation in which the packet is accepted, although we can also reject the packet, leading IPTables to drop it. This figure represents the architecture of our solution for the use case where the security library is used inside our bridging device, mainly to ensure security properties on communications between legacy devices. However, our security library could be use in a standalone mode, and be directly integrated in real IEDs (as we will evaluate in Section 6.3), being a solution capable of ensuring security properties for both legacy or last generation IEDs.

6. Evaluation

To validate our solution and implementation, we need to evaluate in terms of functionality and performance, comparing the obtained results with the require-

ments explained in Section 4. In brief, our solution must ensure the security properties of authentication, integrity and encryption, and at the same time meet the latency requirements expressed in the Table 1, more precisely the 3ms of Transfer Time.

6.1. Evaluation Strategy

With this in mind, our evaluation strategy was divided in two parts, the first one to evaluate the functional requirements and the second to evaluate the performance of each component. We focused our experiments in two components: the security library itself and the Raspberry Pi Bridge. The purpose of evaluating the Security Library by itself was to analyse its impact when incorporated in an IED or other dedicated device. To perform such evaluation we considered two scenarios: first we ran our library only on the Raspberry Pi (specifications on Table 4) to evaluate its performance when running on our own device (lab testbed), secondly we evaluated the performance of the cryptographic algorithms using real IEDs and real R-GOOSE traffics as input.

For the first scenario (lab testbed), we have set up a prototype experiment to simulate the communications between a node of a Power Distribution System and any other node on the network. This experiment is composed by two PCs (specifications on Table 5) representing endpoint nodes of the network and our Raspberry Pi. All of the components are connected by Ethernet cables. One of the nodes will act as a R-GOOSE publisher, while the second will act as a R-GOOSE subscriber, representing two power distribution system nodes. Although as it has fewer nodes it will allow us to analyse the impact that our device will have on the machine-to-machine communications.

The Functional Evaluation phase was only performed on our lab testbed using the Raspberry Pi and we performed the following experiments:

1. **Cryptographic Functions** - Provide a given input based on standardized test vectors from RFCs (RFC 4231 [14], RFC 7693 [15] and NIST SP 800-38D [16]), to the cryptographic function and analyse the output, comparing with the expected. All of the developed functions were tested.

Table 5: PC A and PC B Technical Specifications.

Component	PC A	PC B
CPU	Intel E6400 @ 2.13GHz	i7-6500U @ 2.50GHz
RAM	4GB	16GB
Ethernet	Gigabit Ethernet	Gigabit Ethernet
Operating System	Ubuntu 16.04 (Xenial)	Windows 10

2. **Protocol Related Functions** - Provide an unsecured R-GOOSE packet and analyse the output. On the output, we analysed if the structure and the mutable fields were properly updated, having in consideration the protocol specifications. All of the developed functions were tested using only one cryptographic function. The unsecured R-GOOSE packets were generated by a modified version of libIEC61850 [17].

As mentioned before, for the performance, we evaluated our solution in two scenarios: on our lab testbed composed by COTS components, and on real IEDs using real R-GOOSE traffic. For the performance evaluation phase using our lab testbed we performed the following experiments:

1. **Cryptographic Functions** - Set a timer before the function execution, provide an input to the cryptographic function and set a timer at the end of its execution, measuring the difference between them. All of the developed functions were tested. Due to the fact that the threat where our function is executing can be interrupted, it is necessary to run each experiment multiple times, to get statistically valid results.
2. **Protocol Related Functions** - Set a timer before the function execution, provide an unsecured R-GOOSE packet to the function and set a timer at the end of its execution, measuring the difference between them. All of the developed functions were tested, using all of the cryptographic functions developed.
3. **Raspberry Pi Bridging Device** - To properly evaluate the bridging

device performance, we evaluated the following metrics:

(a) Communications Bandwidth in Mbits/sec

(b) Traffic Latency in milliseconds

4. **Security Gateway** - Generate R-GOOSE packets on the experimental scenario and analyse the latency of communications using the Security Gateway

   - R-GOOSE packets Latency - Only tested the best and worst performing cryptographic functions, using the values measured from the protocol related functions in point 2 above. The R-GOOSE traffic was generated with *libIEC61850* library. Each packet was filled with random data and sent at the highest R-GOOSE rate.

When evaluating the performance of the Security Library, we repeated each experiment 500 000 times, in order to present statistically valid results. From that dataset we calculated the average latency, the standard deviation, maximum and minimum value, and the 95% confidence interval. Also, we considered three input sizes to cover several operation scenarios. Those data sizes were obtained by analysing several R-GOOSE messages. For the Security Library testing, when evaluating the cryptographic algorithms, we used different input sizes for authentication and encryption because in a R-GOOSE packet the amount of data that is authenticated is not the same that is encrypted. The input sizes used are detailed in Table 6, being 196, 256 and 572 bytes for authentication related experiments, and 51, 204 and 408 for encryption/decryption related experiments. To perform authentication and encryption/decryption of R-GOOSE packets, it is used the GOOSE PDU and a set of fixed size fields. To obtain this input sizes, we calculated the total size in bytes of fixed fields that are used on the authentication and for encryption/decryption of an R-GOOSE packet. Analysing several R-GOOSE packets generated by the *libIEC61850*, integrated with several changes performed by Diogo S. et al. in [22], we obtained the size of the GOOSE PDU, when a packet (GOOSE PDU) is composed by 1, 12 or 72 GOOSE objects/entities (integers). We used three different GOOSE PDU

compositions to evaluate the performance of our solution while incrementing the size of the data that is being transmitted. We have evaluated all the security functions implemented and that are part of the security library.

Table 6: Input sizes used for authentication and encryption/decryption related experiments.

Security Mechanism	Input size (bytes)	GOOSE Objects
Authentication	196	1
	256	12
	572	72
Encryption / Decryption	51	1
	204	12
	408	72

When evaluating the Raspberry Pi Bridging Device, we used iPerf3 [25] to measure the bandwidth of the communications with and without the usage of the bridging device. For the latency analysis, we used the Ping tool and we analysed the latency measured using ICMP packets sent at the highest rate used by R-GOOSE, that is 4ms between each packet according to IEC 61850. If our device is capable of handling the traffic at its highest rate, we can assume that it will be able to handle lower rates.

As mentioned before, we also evaluated the library performance on a scenario with a complete product running a real world PDS solution, composed by real IEDs. Our library and in particular the cryptographic functions, run directly inside real IEDs competing for the same computacional resources. There were collected 20 measurements for the latency for each cryptographic function, and three different IED platforms were used. Table 7 illustrates the specifications of the IED platforms used. Finally, we will present a comparison between the latency results for our library, with the results presented in related works.

On the following subsections, we will present the results for the experimental evaluation. Subsection 6.2 presents the results from the evaluation performed on the lab testbed, using COTS hardware components, namely the Raspberry

Table 7: IED platforms specifications.

Platform	Processor	Frequency (MHz)
Platform 1	ARM926EJ-S core	667 MHz
Platform 2	PowerPC e300 core	456 MHz
Platform 3	ARMv7-A core	800MHz

Pi. Subsection 6.3 presents the results from the evaluation performed using real IED devices.

6.2. Evaluation Results from lab testbed

For the functional evaluation of the Security Library, all of the tests passed successfully, both the cryptographic functions analysed using the standardized vectors, as well as the protocol related functions analysed by comparing with the R-GOOSE standard and protocol specification.

In the Table 8 we present the statistic analysis performed on the cryptographic algorithms of the Security Library. From the table, we can see that all of the calculated averages of the cryptographic algorithms latency meet the requirements set for our experiments. We can see that the results are as expected, where GMAC generation is faster than HMAC, but authentication has a better performance than encryption. Figure 4 illustrates the evolution of each authentication related function when increasing the input size. In this figure, we can clearly see that GMAC variants had a better performance than HMAC. Also, HMAC-BLAKE2b had the worst performance, with an higher latency value than any other algorithm. Figure 5 illustrates the evolution of each encryption/decryption related function when increasing the input size. From this plot, we can clearly see the decrease of the performance in encryption when the input size increases.

As for the Raspberry Pi Bridging Device evaluation, we measured and analysed two metrics to evaluate the performance of the bridging device: bandwidth and latency of communications. As for the bandwidth test, we first measure it

Table 8: Security functions performance analysis, in milliseconds, using medium size data input.

Algorithm	**Avg.**	**Max.**	**Min.**	**C.I. 95%**
HMAC-SHA256-80	0.007	0.102	0.007	± 0.0000009
HMAC-BLAKE2b-80	0.012	0.157	0.012	± 0.0000014
GMAC-AES256-128	0.004	1.103	0.004	± 0.0000056
GMAC-AES256-64	0.005	0.218	0.005	± 0.0000019
AES256-GCM-Encrypt	0.032	1.402	0.011	± 0.0000310
AES256-GCM-Decrypt	0.031	0.352	0.011	± 0.0000287

without the bridging device and we collected 1800 measurements for statistical validity. Then, we repeated the experiment but using the bridging device. The results are presented in Table 9.

Table 9: Total data transferred and Bandwidth measurements with and without the bridging device in place.

Without	Average Bandwidth	94.60 Mbits/sec
	Confidence Interval 95%	± 0.05 Mbits/sec
With	Average Bandwidth	94.34 Mbits/sec
	Confidence Interval 95%	± 0.15 Mbits/sec

We can see that bandwidth decreased from 94.60 Mbits/sec, when Raspberry Pi is not in place, to 94.34 Mbits/sec when it is in place. We can conclude that our bridging device does not significantly decrease the bandwidth of the network used.

To measure the latency that our bridging device introduces on the communications we used the well known Ping tool. We used the setup as it is shown in Figure 6 and we executed the Ping command from PC B. The Ping tool measures the RTT of an ICMP packet in a given network. In terms of latency, our final goal is to analyse if our Security Gateway can apply the security

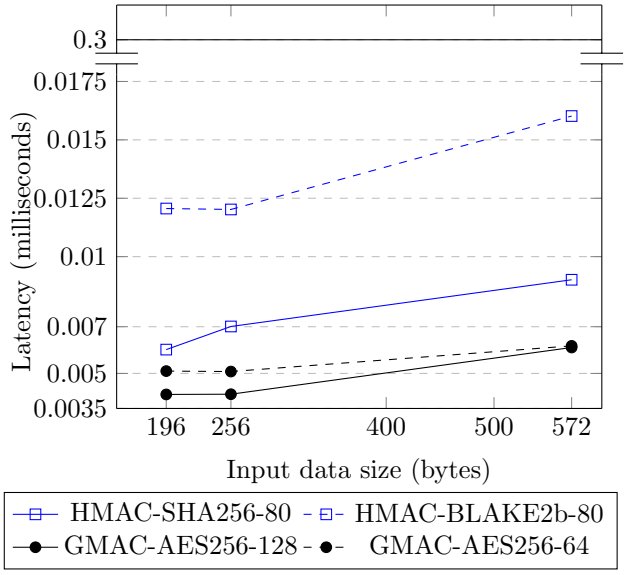

Figure 4: Progression of latency with input size in HMAC and GMAC functions to authenticate data.

mechanisms developed without compromising the 3ms of latency per R-GOOSE packet. This 3ms is the Transfer Time, as illustrated in the Figure 2. There we can see that it is an OTT (One Time Trip) and does not include the latency introduced by the endpoint applications (R-GOOSE Applications).

Given that and the fact that Ping measures the RTT, we can define our objective for this test as achieving an RTT less than 6ms. If we assume that both trip times on Ping calculation are very similar (as they use the same network and packet size are very similar), we can assume that Transfer Time will be less than half of the RTT calculated using Ping, because Transfer Time does not include the processing time on each application.

From the Ping experiment we can confirm that our bridging device complies with the requirements for communications, more precisely, with the 3ms for Transfer Time. The average for RTT of ICMP packets was of 2.437ms. During the experiment we measured the latency of 4 000 000 packets, giving a confidence interval 95% of 0.0441ms. From there we can estimate an OTT of 1.2185ms.

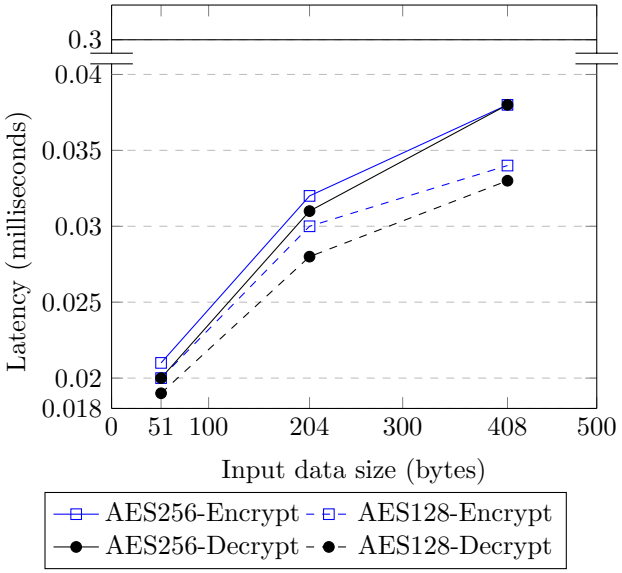

Figure 5: Progression of latency with input size in AES-based functions to encrypt and decrypt data.

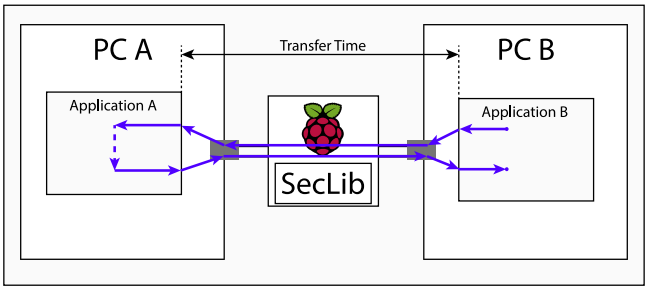

Figure 6: Diagram of Ping testing and relation with Transfer Time.

In a second experiment, we needed to evaluate how the bridge would perform using R-GOOSE packets. The trivial way to achieve this would be to timestamp the messages in each end of the communication and compare both values. In order to be able to precisely measure such latency in this way, the clocks at both ends should be properly synchronized. Although, using only COTS hard-

ware (as a Raspberry Pi), it can be difficult to achieve high precision in clock synchronization due to the very low magnitude of the values being measured.

To overcome this issue, we measured instead the time from the issuer (PC B in Figure 6) to the receiver (PC A) and back (from PC A to PC B), therefore eliminating clock synchronization issues. Since we are considering a simple, fixed and stable network scenario, we then assume that the one-way latency (from PC B to PC A) will be sensibly 1/2 of this round-trip latency, being these the results we later present.

Based on this reasoning, PC A was running an UDP server that receives the R-GOOSE packet and re-sends it to PC B, that is running an R-GOOSE application, calculating the two-way latency (RTT) of the communication. From this RTT we estimate the one-way latency of the communication allowing us to analyse the cost of Transfer Time, represented in Figure 6.

In these tests, we only used the best and worst performing algorithms from the performance evaluation done on our Security Library. These tests were focused on analysing the latency on R-GOOSE communications passing through the Security Gateway. As mentioned before, to measure latency, we developed an application that was acting as an UDP/R-GOOSE Server. This application was placed in PC A and was receiving R-GOOSE packets sent from PC B. On PC B, packets were generated and sent using our modified version of libIEC61850 [17].

As we did in the analysis of Ping results for bridging device, in these tests we are also measuring the RTT and not the Transfer Time. Although, as we showed on Figure 6, we can estimate the OTT (that is always bigger than Transfer Time of Figure 2) and evaluate if the test was successful or not. From the Table 10 we can see that all of the RTT values are between 3.046 ms and 3.207 ms, from where we can estimate an OTT between 1.523 ms and 1.6035 ms. With this OTT values, we can state that our Security Gateway complies with the R-GOOSE requirement of a Transfer Time minor than 3ms, as per the requirements previously discussed.

Table 10: Bridge device latency (RTT) measurements in milliseconds.

Name	Avg	CI 95%	Max	Min
AddHMAC-SHA256-128	3.165	0.033	8.126	1.829
AddHMAC-BLAKE2b-80	3.122	0.033	7.687	2.315
AddGMAC-AES128-128	3.100	0.036	12.148	2.266
AddGMAC-AES256-128	3.121	0.035	7.011	2.231
ValidateHMAC-SHA256-80	3.155	0.045	10.75	2.289
ValidateHMAC-BLAKE2b-80	3.073	0.045	11.234	2.273
ValidateGMAC-AES128-128	3.184	0.043	11.205	2.28
ValidateGMAC-AES256-64	3.051	0.044	10.067	2.237
Encrypt-AES128	3.069	0.037	8,688	2.238
Encrypt-AES256	3.207	0.038	12.689	2.41
Decrypt-AES128	3.129	0.047	11.661	2.162
Decrypt-AES256	3.046	0.034	12.989	2.243

6.3. Evaluation Results from real IEDs testbed

In this subsection we will present the results and analysis performed to evaluate the latency of cryptographic algorithms from our security library, when running on real IED devices (see Table 7) with real R-GOOSE traffic.

On Tables 11, 12 and 13, we present the latency results for the the cryptographic functions respectively using the IED platforms 1, 2 and 3 as specified in Table 7.

These results support the evaluation we did on the lab testbed. We can see that in platforms 2 and 3 (that are the most capable IEDs) all of the algorithms except HMAC-BLAKE2b had an acceptable performance, what supports the results we obtained from the experiments performed on our testbed. Comparing both experiment approaches, we can see that the latency values are higher in real devices, as it was expected given the fact that our Raspberry Pi is more capable than the real IEDs used on these experiments. From this analysis,

Table 11: Latency results for the IED Platform 1 of Table 7 (in milliseconds).

Cryptographic algorithm	Avg	CI 95%	Max	Min
GMAC-AES128-64	0.242	0.024	0.435	0.223
GMAC-AES128-128	0.241	0.019	0.398	0.227
GMAC-AES256-64	0.226	0.015	0.328	0.214
GMAC-AES256-128	0.228	0.028	0.453	0.206
HMAC-SHA256-80	0.276	0.289	0.439	0.235
HMAC-SHA256-128	0.245	0.172	0.345	0.226
HMAC-SHA256-256	0.272	0.053	0.641	0.221
HMAC-BLAKE2b-80	2.228	0.037	2.476	2.11
HMAC-BLAKE2s-80	0.363	0.035	0.614	0.321
AES256-GCM-Encrypt	0.401	0.046	0.678	0.341
AES128-GCM-Encrypt	0.374	0.039	0.641	0.326
AES256-GCM-Decrypt	0.385	0.040	0.674	0.344
AES128-GCM-Decrypt	0.363	0.044	0.793	0.332

we can state that for platforms 2 and 3 the cryptographic algorithms can be directly applied on IEDs. When using platforms less capable such as platform 1, the computational cost to perform such cryptographic operations is higher, meaning that an external device, as our proposed bridging device, should be used to comply with the latency restrictions.

On Table 14 we present all of our results and also results obtained in related work [30]. From this table, we can assess the impact of different security mechanisms on several platforms. Among our real IED platforms, we can conclude that there are some platforms where security mechanisms can be directly applied (IED Plat. 2 and 3), while IED Plat. 1 is not capable of supporting such mechanisms.

Our results obtained using RPi 4B are similar to the results obtained in [30], except for the encryption algorithms. However, for both authentication and

Table 12: Latency results for the IED Platform 2 of Table 7 (in milliseconds).

Cryptographic algorithm	Avg	CI 95%	Max	Min
GMAC-AES128-64	0.059	0.015	0.196	0.048
GMAC-AES128-128	0.060	0.022	0.229	0.038
GMAC-AES256-64	0.062	0.016	0.187	0.04
GMAC-AES256-128	0.058	0.013	0.149	0.045
HMAC-SHA256-80	0.083	0.016	0.197	0.065
HMAC-SHA256-128	0.059	0.009	0.142	0.052
HMAC-SHA256-256	0.062	0.010	0.156	0.052
HMAC-BLAKE2b-80	1.123	0.015	1.196	1.084
HMAC-BLAKE2s-80	0.087	0.011	0.187	0.077
AES256-GCM-Encrypt	0.130	0.024	0.259	0.096
AES128-GCM-Encrypt	0.094	0.012	0.186	0.078
AES256-GCM-Decrypt	0.106	0.019	0.262	0.086
AES128-GCM-Decrypt	0.090	0.016	0.214	0.074

encryption, we found it difficult to provide a detailed and objective comparison, due to the lack of technical details (input size, number of repetitions) at the experimental level of work [30]. Finally, from this table we can see that even low capability devices as RPi 4B or Intel Celeron(R) 4GB do not represent the most constrained devices present in real world PDS, being important to assess the real impact of security mechanisms on real devices, as our IED platforms.

7. Conclusions and Future Work

We propose a Security Gateway for Power Distribution Systems (PDS), composed of a Security Library and a Bridging Device. The Security Library features the implementation of several cryptographic mechanisms as recommended in corresponding standards for data communications in PDS, namely for authentication, integrity and confidentiality. The Security Library can be used

Table 13: Latency results for the IED Platform 3 of Table 7 (in milliseconds).

Cryptographic algorithm	Avg	CI 95%	Max	Min
GMAC-AES128-64	0.024	0.006	0.076	0.017
GMAC-AES128-128	0.023	0.007	0.094	0.018
GMAC-AES256-64	0.024	0.008	0.086	0.017
GMAC-AES256-128	0.021	0.005	0.054	0.015
HMAC-SHA256-80	0.041	0.005	0.081	0.035
HMAC-SHA256-128	0.034	0.005	0.065	0.027
HMAC-SHA256-256	0.034	0.007	0.097	0.028
HMAC-BLAKE2b-80	0.158	0.030	0.387	0.130
HMAC-BLAKE2s-80	0.044	0.007	0.087	0.037
AES256-GCM-Encrypt	0.053	0.011	0.146	0.043
AES128-GCM-Encrypt	0.045	0.012	0.152	0.036
AES256-GCM-Decrypt	0.051	0.013	0.153	0.037
AES128-GCM-Decrypt	0.043	0.011	0.136	0.034

standalone by the PDS devices, or through a Bridging Device that was developed to enable security features for low-capability/legacy PDS devices. Our results show that it is feasible to use such cryptographic algorithms in PDS, as they provide the required security features while complying with the strict performance restrictions in this environment. Moreover, using our Bridging Device, we demonstrated that is possible to use COTS equipment to secure legacy and less capable PDS devices, also complying with the stringent performance requirements.

As future work, we plan to validate this solution in a larger experimental scenario, using real devices and analysing the impact that our solution has on other system components. Another interesting challenge would be to include in the framework a set of functions to translate from GOOSE packets to R-GOOSE packets, allowing the application of our solution in full legacy systems

Table 14: Comparison with related works

Work	Device	Algorithm	Avg. Latency (ms)
[30]	Intel Celeron(R) 4GB	HMAC-SHA256	0.008
This	RPi 4B	HMAC-SHA256-80	0.005
This	IED Plat. 1	HMAC-SHA256-80	0.276
This	IED Plat. 2	HMAC-SHA256-80	0.083
This	IED Plat. 3	HMAC-SHA256-80	0.041
[30]	Intel Celeron(R) 4GB	AES-GMAC-64	0.0045
This	RPi 4B	GMAC-AES256-64	0.005
This	IED Plat. 1	GMAC-AES256-64	0.226
This	IED Plat. 2	GMAC-AES256-64	0.062
This	IED Plat. 3	GMAC-AES256-64	0.024
[30]	Intel Celeron(R) 4GB	AES-GMAC-128	0.005
This	RPi 4B	GMAC-AES256-128	0.004
This	IED Plat. 1	GMAC-AES256-128	0.228
This	IED Plat. 2	GMAC-AES256-128	0.058
This	IED Plat. 3	GMAC-AES256-128	0.021
[30]	Intel Celeron(R) 4GB	AES-GCM-256-Enc	0.286
This	RPi 4B	AES-GCM-256-Enc	0.032
This	IED Plat. 1	AES-GCM-256-Enc	0.401
This	IED Plat. 2	AES-GCM-256-Enc	0.130
This	IED Plat. 3	AES-GCM-256-Enc	0.053
[30]	Intel Celeron(R) 4GB	AES-GCM-256-Dec	0.221
This	RPi 4B	AES-GCM-256-Dec	0.031
This	IED Plat. 1	AES-GCM-256-Dec	0.385
This	IED Plat. 2	AES-GCM-256-Dec	0.106
This	IED Plat. 3	AES-GCM-256-Dec	0.051

that do not support R-GOOSE. Furthermore, it is extremely important to take in consideration the way that security keys that will be used by our library are exchanged and managed. This is another topic for a future work, where IEC already recommends the usage of Group Domain of Interpretation protocol (GDOI) as key exchange protocol. Finally, as part of our proposal is using an external device to apply the security mechanisms, it is extremely important to ensure the security of the device itself, in this case Raspberry Pi. Given this, it would be important to research on how secure is such device and assess what measures should be taken to protect the device.

Acknowledgements

This work is supported by the European Regional Development Fund(FEDER), through the Regional Operational Programme of Lisbon(PORLISBOA 2020) and the Competitiveness and Internationalisation Operational Programme (COMPETE 2020) of the Portugal 2020 framework[Project 5G with Nr.024539 (POCI-01-0247-FEDER-024539)].

References

[1] M. Daoud and X. Fernando, "On the Communication Requirements for the Smart Grid", Energy and Power Engineering, 3, Jan. 2011.

[2] Ye Yan, Yi Qian, H. Sharif, and D. Tipper, "A survey on cyber security for smart grid communications", Communications Surveys and Tutorials, IEEE, 14:998–1010, Jan. 2012.

[3] A. Hadbah, A. Kalam, and A. Zayegh. "Powerful IEDs, ethernet networks and their effects on IEC 61850-based electric power utilities security", In 2017 Australasian Universities PowerEngineering Conference (AUPEC), pages 1–5, Nov. 2017.

[4] A. Apostolov, "R-GOOSE: what it is and its application in distribution automation", CIRED - Open Access Proceedings Journal, 2017(1):1438–1441, 2017.

[5] International Electrotechnical Commission, "Power Utility Automation", Standard IEC 61850.

[6] International Electrotechnical Commission, "Power systems management and associated information exchange - data and communications security", Standard IEC 62351, 2018.

[7] O. Khaled, A. Marín, F. Almenares, P. Arias, and D. Díaz, "Analysis of secure TCP/IP profile in 61850 based substation automation system for smart grids", In International Journal of Distributed Sensor Networks, p. 5793183, 2016.

[8] S. R. Firouzi, L. Vanfretti, Al. Ruiz-Alvarez, H. Hooshyar, and F. Mahmood, "Interpreting and implementing IEC 61850-90-5 Routed-Sampled Value and Routed-GOOSE protocols for IEEE C37.118.2 compliant wide-area synchrophasor data transfer", Electric Power Systems Research, 144:255–267, Mar. 2017

[9] S. Obermeier, R. Schlegel, and J. Schneider, "Assessing the Security of IEC 62351", Proceedings of the 3rd International Symposium for ICS and SCADA Cyber Security Research 2015, pages 11–19, Jan. 2015.

[10] S. M. Farooq, S. M. Suhail Hussain, and T. S. Ustun, "Performance Evaluation and Analysis of IEC 62351-6 Probabilistic Signature Scheme for Securing GOOSE Messages", IEEE Access, Mar. 2019.

[11] S. Fuloria and R. Anderson, "The Protection of Substation Communications", In SCADA Security Scientic Symposium, 2010.

[12] Eduardo Andrade, "R-GOOSE-SecLib", https://github.com/slipz/R-GOOSE_SecLib. (13/12/2020)

[13] Eduardo Andrade, "RPi-Gateway", https://github.com/slipz/RPi-Gateway. (13/12/2020)

[14] M. Nystrom, "Identifers and Test Vectors for HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512", RFC 4231, RFC Editor, Dec. 2005.

[15] J. Aumasson and M. O. Saarinen, "The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC)", RFC 7693, RFC Editor, Nov. 2015.

[16] National Institute Of Standards and Technology, "NIST Special Publication 800-38D Computer Security - Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST, Gaithersburg, MD 20899-8930, 2007.

[17] MZ Automation, "Oficial repository for libIEC61850, the open-source library for the IEC61850 protocols", github.com/mz-automation/libiec61850. (13/12/2020)

[18] M. A. Elgenedy, A. M. Massoud and S. Ahmed, "Smart grid self-healing: Functions, Applications, and Developments", 2015 First Workshop on Smart Grid and Renewable Energy (SGRE), Mar. 2015, Doha, Qatar.

[19] A. Aleixo, J. Cabaça, P. M. Neves, R. Dias Jorge, R. Dias Paulo and A. Rodrigues, "Smart Grid protection and automation enabled by IEC 61850 communications over 5G Networks", 25th International Conference on Electricity Distribution, Jun. 2019, Madrid, Spain.

[20] International Electrotechnical Commission, "Communication networks and systems for power utility automation – part 90-5: use of IEC 61850 to transmit synchrophasor information according to IEEE C37.118", Standard IEC TR 61850 90-5, 2012.

[21] International Electrotechnical Commission, "Power systems management and associated information exchange - Data and communications security - Part 6: Security for IEC 61850", Draft for Standard IEC 62351-6, 2020.

[22] D. M. Saraiva, D. Corujo and R. L. Aguiar, "IEC 61850 Data Transfer Evaluation over Public Networks", 22nd International Symposium on Wireless Personal Multimedia Communications (WPMC), Nov. 2019, Lisbon, Portugal.

[23] R. Khan, K. Mclaughlin, D. Laverty and S. Sezer, "Design and Implementation of Security Gateway for Synchrophasor Based Real-Time Control and Monitoring in Smart Grid", IEEE Access, 2017(5):11626-11644, 2017

[24] Free Software Foundation (2001), "Man Page of brctl", Available at https://manpages.debian.org/testing/bridge-utils/brctl.8.en.html (17/11/2020)

[25] DAST and NLANR, "iPerf3", Available at https://iperf.fr (19/11/2020)

[26] K. Curtis, "A DNP3 Protocol Primer (Revision A)", DNP3 Users Group, Calgary, Canada (https://www.dnp.org/Portals/0/AboutUs/DNP3%20Primer%20Rev%20A.pdf) (last visited: 27/11/2020)

[27] S. M. S. Hussain, S. M. Farooq and T. S. Ustun, "Analysis and implementation of message authentication code (MAC) algorithms for GOOSE message security", IEEE Access, vol. 7, pp. 80980-80984, 2019

[28] S. M. S. Hussain, S. M. Farooq and T. S. Ustun, "A method for achieving confidentiality and integrity in IEC 61850 GOOSE messages", IEEE Trans. Power Del., vol. 35, no. 5, pp. 2565-2567, 2020

[29] T. S. Ustun, S. M. Farooq and S. M. S. Hussain, "A Novel Approach for Mitigation of Replay and Masquerade Attacks in Smartgrids Using IEC 61850 Standard", IEEE Access, vol. 7, pp. 156044-156053, 2019

[30] T. S. Ustun, S. M. Farooq and S. M. S. Hussain, "Implementing Secure Routable GOOSE and SV Messages Based on IEC 61850-90-5", IEEE Access, vol. 8, pp. 26162-26171, 2020

[31] M. Rodríguez, J. Lázaro, U. Bidarte, J. Jiménez and A. Astarloa, "A Fixed-Latency Architecture to Secure GOOSE and Sampled Value Messages in Substation Systems", IEEE Access, vol. 9, pp. 51646-51658, 2021

[32] D. Ishchenko and R. Nuqui, "Secure communication of intelligent electronic devices in digital substations", 2018 IEEE/PES Transmission and Distribution Conference and Exposition (T&D). IEEE, Apr. 2018.

[33] N. Kush, M. Branagan, E. Foo, and E. Ahmed, "Poisoned GOOSE : exploiting the GOOSE protocol", Conferences in Research and Practice in Information Technology Series, 149, 01 2014.

[34] J. Hoyos, M. Dehus, and T. X. Brown, "Exploiting the GOOSE protocol: A practical attack on cyber-infrastructure", 2012 IEEE Globecom Workshops, pages 1508–1513, Dec 2012.

[35] J. G. Wright and S. D. Wolthusen, "Stealthy injection attacks against IEC61850's GOOSE messaging service", 2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe). IEEE, Oct. 2018.

[36] R. P. Gupta, “Substation automation using iec 61850 standard”, Fifteenth National Power Systems Conference (NPSC), IIT Bombay, Dec. 2008.

[37] A. Apostolov, ”R-GOOSE: what it is and its application in distribution automation”, CIRED - Open Access Proceedings Journal, 2017(1):1438–1441, 2017.