applied sciences

MDPI

Article

Analysis of Cryptographic Algorithms to Improve Cybersecurity in the Industrial Electrical Sector

Francisco Alonso [1], Benjamín Samaniego [2], Gonzalo Farias [2] and Sebastián Dormido-Canto [1,*]

[1] Departamento de Informática y Automática, Universidad Nacional de Educación a Distancia, 28040 Madrid, Spain; falonso332@alumno.uned.es
[2] Escuela de Ingeniería Eléctrica, Pontificia Universidad Católica de Valparaíso, Valparaíso 2362804, Chile; benjamin.samaniego.e@mail.pucv.cl (B.S.); gonzalo.farias@pucv.cl (G.F.)
* Correspondence: sebas@dia.uned.es; Tel.: +34-91398-7194

Abstract: This article provides a general overview of the communication protocols used in the IEC61850 standard for the automation of electrical substations. Specifically, it examines the GOOSE and R-GOOSE protocols, which are used for exchanging various types of information. The article then presents real cases of cyber attacks on the industrial sector, highlighting the importance of addressing cybersecurity in the IEC61850 standard. The text presents security drawbacks of the communication protocols mentioned earlier and briefly explains two algorithms defined in the IEC61850 standard to address them. However, the authors suggest that having only a couple of algorithms may not be sufficient to ensure digital security in substations. This article presents a study on the cryptographic algorithms ChaCha20 and Poly1305. The purpose of the study is to experimentally verify their adaptation to the strict time requirements that GOOSE must meet for their operation. These algorithms can operate independently or in combination, creating an Authenticated Encryption with Associated Data (AEAD) algorithm. Both algorithms were thoroughly reviewed and tested using GOOSE and R-GOOSE frames generated by the S-GoSV software. The computational time required was also observed. The frames were analysed using the Wireshark software. It was concluded that the algorithms are suitable for the communication requirements of electrical substations and can be used as an alternative to the cryptographic algorithms proposed under the IEC61850 standard.

Keywords: IEC61850; IEC62351; Generic Object-Oriented Substation Events (GOOSE); R-GOOSE; ChaCha20; Poly1305; AEAD

Citation: Alonso, F.; Samaniego, B.; Farias, G.; Dormido-Canto, S. Analysis of Cryptographic Algorithms to Improve Cybersecurity in the Industrial Electrical Sector. *Appl. Sci.* **2024**, *14*, 2964. https://doi.org/10.3390/app14072964

Academic Editor: Anyang Lu

Received: 21 February 2024
Revised: 28 March 2024
Accepted: 29 March 2024
Published: 31 March 2024

1. Introduction

Technological fields have evolved exponentially in recent decades. In the area of electrical power systems (EPSs), the basic structure of the electricity network was maintained for more than 80 years without significant changes [1]. Less than two decades ago, the old substations, automated through point-to-point copper connections, were migrating towards fiber optic communication networks through Ethernet LAN networks and also through WAN networks for communication between distant substations. Fiber optic communications use an infrastructure made up of network equipment, such as routers, switches, servers, etc. This technology is commonly called ICT (information communication technology). When ICT is applied to any electrical power system, the concept of a "smart grid" is born. IEC61850 [2] is an international standard which defines different communication protocols between various field equipment located in automated substations. It was developed by the International Electrotechnical Commission (IEC) in 2003 with the intention of modernizing and optimizing the automation of substations so that they are suitable for the energy demands of the 21st century. Furthermore, to achieve interoperability between hardware from different suppliers, different standardized communication protocols were

developed, preventing each supplier from having their internal protocols and causing any incompatibility between equipment.

The communication protocols of the IEC61850 standard define how information is exchanged between the different levels of automation within the substation. The information from field-level equipment, such as switches, disconnectors, and transformers, is digitized through devices called merging units [3]. The exchange of information in IEC61850 is subject to demanding latency times that must be met. The design of the different functions and equipment within the substation is object-oriented, which means that each element or function is described in a well-defined way to achieve the desired interoperability between hardware from different vendors. IEC61850 has the ability to report many relevant data and much information almost instantly. Thanks to these qualities, its object-oriented design approach, and its interoperability features, IEC61850 has become by far the most popular standard for the automation of electrical energy services and not only for substation automation systems, for which it was invented, but also for other areas of smart network communication such as those existing in electric vehicles (EVs) [4]. However, these great qualities and attributes of the IEC61850 standard are diminished due to the vulnerabilities to cyber attacks that originate when using standardized communication protocols that use equipment such as LAN switches and routers that can be exposed to a public network such as the Internet. Standardized semantics make it much easier for a cyber attacker to launch different types of attacks that can be catastrophic for the operation of the substation [5,6]. This is why various types of attacks have been carried out on these systems, which have generated considerable operational, economic, and even human losses [7].

The paradigm that cybersecurity problems would not affect electrical power systems had to change. In 2007, WG15 launched the IEC62351 standard, which complements the IEC61850 standard and other standards under the jurisdiction of TC57 by proposing different cybersecurity methodologies using cryptographic algorithms [8]. IEC62351 has successfully prevented different attacks on the IEC61850 communication protocols [9].

The investigative work requires the development of different IEC61850 communication models, their implementation in a laboratory environment, the exchange of real information using the communication protocols of the standard, and observing the communication performance to study the impact on the operation of the electrical power system.

Since GOOSE messages have a 3 ms delivery time as a priority requirement, IEC 62351-1 specifies that encryption algorithms should not be used, as the processing times for encrypting GOOSE messages would be longer due to the limited computing capacity of intelligent electronic devices (IEDs). In addition, the standard does not specify message confidentiality as a fundamental requirement. However, GOOSE messages contain sensitive information related to the electricity market or energy administration, which would create a serious management problem if these messages were obtained and decrypted by attackers. Therefore, given this new perspective, GOOSE messages need to be encrypted to add the confidentiality requirement but still meet the 3 ms timing requirement.

Based on the above, the main contribution of this article is to provide a comprehensive investigation of the ChaCha20 and Poly1305 algorithms, both individually and together as components of an AEAD algorithm, to achieve the confidentiality and integrity of GOOSE and R-GOOSE messages, which in turn meet the stringent requirements of the IEC61850 standard, thus establishing them as valid alternatives to the AES algorithm specified in the IEC62351-1 standard, an algorithm that has been thoroughly studied by various cryptanalyses, which is currently the only recommended algorithm in the standard.

Through experimental validation, the paper shows its ability to meet the strict timing constraints of the IEC61850 standard for frame transmission, specifically in the context of the GOOSE and R-GOOSE protocols. As a result, this study positions these algorithms as credible alternatives to the existing ones described in the standard.

The structure of this article unfolds as follows: Section 2 provides details of the GOOSE and R-GOOSE communication protocols, shedding light on the prevailing cybersecurity challenges within the IEC61850 standard. Section 3 delves into fundamental cybersecurity

concepts, examining notable attacks targeting the aforementioned protocols, and scrutinizing the cryptographic algorithms endorsed in the IEC62351 standard to fortify the sector against diverse threats. Lastly, Section 4 meticulously expounds on the algorithms proposed herein—ChaCha20 and Poly1305—alongside their experimental validation, either in isolation or in tandem as an AEAD algorithm. This section meticulously demonstrates their ability to adhere to the stringent time constraints mandated by the IEC61850 standard for transmitting GOOSE and R-GOOSE frames. Furthermore, it includes a comprehensive comparative analysis between the proposed algorithms and those stipulated in the standard.

2. Communication Protocols GOOSE and R-GOOSE and Disadvantages of the IEC61850 Standard

Within the electrical substations under the standard, the field level is distinguished, where the voltage and current transformers, disconnectors, switches, and merging units that digitize the voltage and current values of the transformers are located. The bay level, where the IEDs are located, applies control and protection over the field-level equipment and sends reports to the next higher level, the station level. There are three distinct types of messaging or information exchange, including GOOSE, SV, and MMS. The other two, R-GOOSE and R-SV, are used when this messaging must be sent to other networks [10,11]. The protocol stack of IEC61850, shown in Figure 1, contains the different types of messages mapped to different layers of the OSI model.

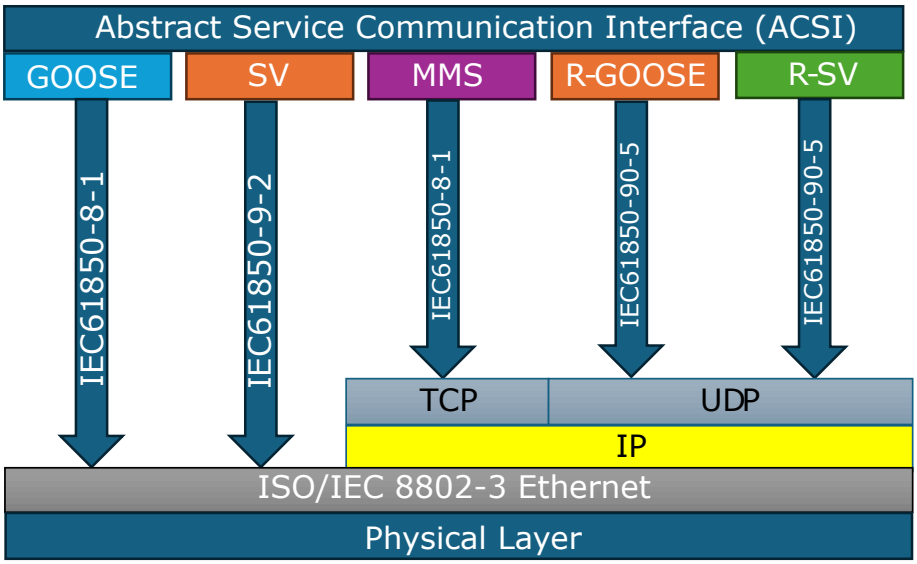

Figure 1. Mapping of the communication protocols of the IEC61850 standard.

2.1. GOOSE

GOOSE (Generic Object-Oriented Substation Events) allows bay-level equipment (IEDs) to exchange information with each other if they are within the same network through a publisher–subscriber configuration using a multicast system, where a selected group of IEDs (subscribers) receive all the information simultaneously and act if required. This information comes from another IED called a publisher. GOOSE carries out the exchange of information between the process level and the bay level since if a failure occurs the IEDs must send GOOSE messages to the elements that must clear the failure. Due to the low transmission latency required by the GOOSE messages of 3 ms maximum, they are mapped directly to the link layer of the OSI model, which avoids the overhead of higher levels of network abstraction [12].

2.2. R-GOOSE

With the development of smart grids, the concept of WAMPAC (wide area monitoring, protection and control) applications was developed. WAMPAC makes use of synchrophasor technology. A synchrophasor is the calculation of a phasor concerning an absolute time reference. With this measurement, the absolute relationship between the phases (angles) of phasors in different locations of the electrical power system can be determined. The time reference is usually a Global Positioning System (GPS). WAMPAC is based on PMUs (phasor measurements units) and PDCs (phasor data concentrators).

PMUs are units of synchrophasor measurements that allow the observation of the static and dynamic behavior of an electrical power system, through the variables of current (amplitude and angle), voltage, frequency, and rate of change of frequency (ROCOF), and are then sent to PDCs through a communication network. There are two PMU communication frameworks: IEEE C37.118 and IEC61850-90-5. To achieve compatibility between IEEE C37.118-based synchrophasor data transfer with the IEC61850 substation automation standard, IEC61850-90-5 was introduced [13,14].

Since synchrophasor-based technology normally requires transmitting data over a WAN, IEC61850-90-5 specifies the use of UDP as a transport protocol, and thus, the R-GOOSE and R-SV messaging type was born. IEC61850-90-5 also defines the digital security requirements that R-GOOSE and R-SV must have, indicating that authentication and integrity of messages are mandatory, while confidentiality (encryption) is optional. This point is very relevant because IEEE C37.118 does not specify cybersecurity requirements to protect data communication over an insecure network, which is supplemented by IEC61850-90-5.

Synchrophasor technology plays a key role in the monitoring, control, and protection of electrical power systems and any failure in this field can have serious consequences such as blackouts. To achieve authentication and integrity of R-GOOSE and R-SV messages, IEC61850-90-5 specifies the use of Message Authentication Code (MAC) algorithms, such as HMAC-SHA256 or AES128-GMAC. Although confidentiality is optional, the technical report specifies AES-128 and AES-256 algorithms as an encryption method to ensure confidentiality. Figure 2 shows an R-GOOSE/R-SV message, including fields with digital security specifications that implements IEC61850-90-5 [15].

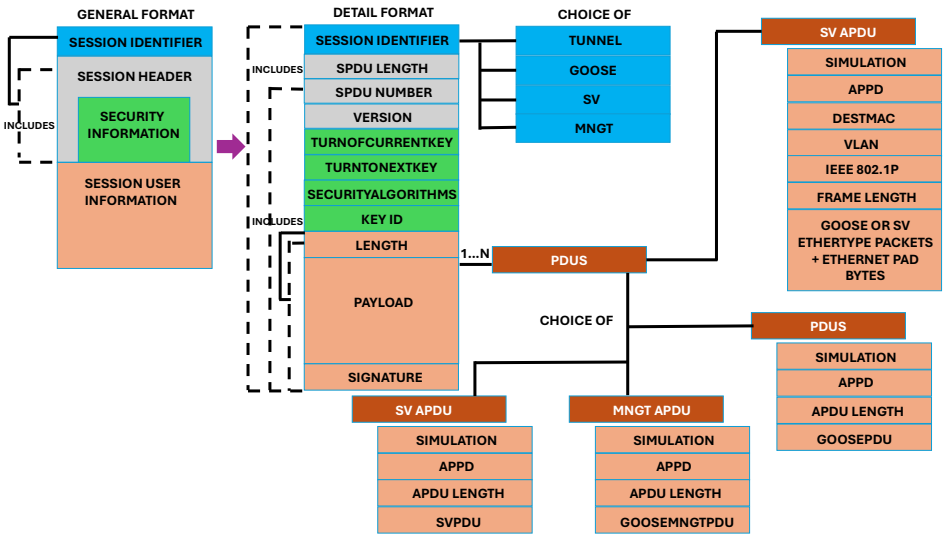

Figure 2. Session PDU field of an R-GOOSE/R-SV frame.

2.3. Disadvantages of the IEC61850 Standard

Although the IEC61850 standard offers numerous advantages, it does not cover an important aspect: cybersecurity. This aspect is generally not relevant to those working in electrical substations or the power systems field, particularly in Latin America where the majority of substations are still conventional.

However, due to the standardization of communication protocols, staff can no longer ignore the threat of cyber attacks. Attacks on electrical systems already exist, as demonstrated by the Stuxnet virus. This virus infected a large number of computers worldwide through pen drives, with the specific objective of infecting Siemens PLCs of a particular model that controlled centrifuges responsible for stirring uranium in the form of gas to enrich it. The ultimate goal was assumed to be the creation of nuclear weapons. It is important to note that this information is presented objectively and without any subjective evaluations. It is estimated that the virus caused damage to over a thousand centrifuges by altering the speed of their motors [7].

Another example of a cyber attack on the electrical industry is Dragonfly, which specifically targets industrial control systems (ICSs) used in the energy sector. This advanced persistent threat (APT) had a significant impact on European countries in 2014. Dragonfly uses several remote access tool (RAT) components to infect and remotely control the affected computers [16].

In 2017, Ukraine was hit by another malware called CrashOverride. This malware targeted organizations that use IEC101, IEC104, IEC61850, and OPC protocols. It has modules that are specific to ICS protocol stacks, as well as non-ICS-specific modules, such as a Wiper that removes files and processes from the running system. The malware exploits switches driven by RTUs and IEDs, forcing them into an infinite loop and keeping circuit breakers open, even if operators try to close them [17].

Figure 3 shows the malware that has been found in different industrial control systems, with emphasis on CrashOverride, which was made especially to attack power networks that use IEC61850 communication protocols [18].

Figure 3. Malware with attacks on industrial control systems.

3. Cybersecurity in the IEC61850 Standard

Information security is defined as the set of preventive and reactive measures of organizations and technological systems that allow for safeguarding and protecting information, seeking to maintain the reliability, availability, and integrity of the data.

In the cybersecurity community, the acronym "CIA" stands for confidentiality, integrity, and accessibility, also known as the CIA triad. It is important to define these three concepts to have a general notion of what cybersecurity seeks:

- Confidentiality : Property that prevents the disclosure of information to unauthorized individuals, entities, or processes. Ensures access to information only to those people who have proper authorization;
- Integrity: Property that seeks to keep the data free of unauthorized modifications, that is, integrity seeks to accurately maintain the information as it was generated, without being manipulated or altered by unauthorized people or processes;
- Availability: Property that ensures that networks, systems, and applications are fully operational. Ensures that users have timely and reliable access to resources when they need them.

Some other relevant concepts in the field of cybersecurity are:

- Authenticity: The sender of the message is who they say they are;
- Non-repudiation of origin: The sender cannot deny what they sent because the recipient has proof of the sending;
- Destination non-repudiation: The receiver cannot deny that they received the message because the sender has proof of receipt.

To achieve information security and ensure that these concepts are carried out correctly, one of the most effective ways is the use of algorithms based on cryptography. A cryptography algorithm modifies the data of a document to achieve some of the characteristics mentioned above, such as authentication, integrity, and/or confidentiality.

One of the most used algorithms currently, and proposed by IEC62351 for the R-GOOSE/R-SV protocols, to guarantee confidentiality, due to its high level of security with attacks, is AES (Advanced Encryption Standard). AES is a symmetric cryptography algorithm which uses the same key to encrypt and decrypt messages between a sender and a receiver. The two parties must agree in advance on the key to use. AES has a fixed block size of 128 bits and key sizes of 128, 192, or 256 bits. Most of the AES algorithm calculations are performed in a given finite field, called the Galois field [19–22].

To ensure integrity and authenticity, digital signature cryptographic algorithms and MAC (Message Authentication Code) cryptographic algorithms are used. Digital signatures employ asymmetric cryptography algorithms (public–private key pair) and pseudo-random hash algorithms, while MAC algorithms use symmetric cryptography algorithms (the same key to generate and verify the authenticity and integrity of messages) and pseudo-random hash algorithms. A hash function is a mathematical function that converts an input string of any length into a fixed-size output, which is a pseudo-random representation of the input. Even a small change in the input string will result in a completely different output hash. If the same input string is used again, the output of the hash function will always be the same, which appears to be random. Hash functions can be thought of as the digital fingerprint of the message. SHA256 is one of the most commonly used pseudo-random algorithms today. It belongs to the SHA-2 (Secure Hash Algorithm) family. The SHA256 algorithm, developed by the National Security Agency (NSA), processes 512-bit message blocks by performing various operations such as addition, logical AND, logical OR, exclusive OR, shift right, and rotate within an internal state of bits. The number of these bits is equal to the number of output bits of the function. These operations are performed 64 times (iterations or loops) in the case of SHA256, resulting in a 256-bit string as output.

The HMAC algorithm ensures message authenticity and integrity by using a hash algorithm, such as SHA256, and a symmetric key. The output of the HMAC algorithm is a fixed string value that represents the message's authenticity and integrity [23].

3.1. Vulnerabilities in the GOOSE Protocol

Cyber attacks on GOOSE messaging can cause catastrophic problems within an electrical substation, for example, irreparable damage to field equipment such as switches or disconnectors. GOOSE messaging works at layer 2 of the OSI model and there are several known attacks for this layer in particular, such as STP attacks, MAC spoofing, CAM table overflows, and VLAN hopping. Below are some attacks that can be performed on GOOSE messaging:

- stNum attack: After inspecting GOOSE messages, malware proceeds to send a GOOSE message with a high number of stNum to the subscribing IEDs. They will think that an event has occurred in the substation and will proceed to process the false GOOSE message.
- GOOSE flood attack: Once the malware manages to obtain a GOOSE frame, it floods the network by sending in a very short period the same GOOSE frame that it obtained, but with the APDU field modified and filled in such a way as to reach the maximum supported by the GOOSE frame (approximately 1500 bytes). This attack has the purpose of compromising the processing capacity of the IEDs and the network.

- Semantic spoofing GOOSE attack: After encountering a GOOSE message on the network, reproducing the original message, but with false information, the malware checks and manipulates the boolean (for example, the state of a switch) or analog information of the APDU. The malware also simulates message transition mechanics by incrementing stNum and zeroing sqNum. The intention is to deceive the IED through a false message.
- Goose replay attack: It uses the injection of an old but real GOOSE message from the substation having an event, for example, an overcurrent fault, through some open port. This open port could be an unconfigured switch port or a test port with access to all network traffic.

The GOOSE flood attack was performed by [6,12] , and the stNum attack and semantic spoofing GOOSE attack by [12], where all attacks on GOOSE messaging were successfully performed. More details of the semantic spoofing GOOSE attack can be reviewed in [9].

3.2. IEC 62351-6: Security for GOOSE

The challenges of securing GOOSE and SV messaging, by verifying message authentication, integrity, and encryption, are much more difficult to achieve than in other systems, since these protocols differ greatly in network latency compliance requirements.

In 2007, WG15 developed the IEC62351 standard that addresses cybersecurity issues of power system communication standards under the jurisdiction of TC57, such as IEC61850. IEC62351-1 stipulates that GOOSE and SV messages must have two mandatory security measures, the authenticity and integrity of the messages. While for confidentiality, IEC62351-1 stipulates that encryption algorithms cannot be applied due to the difficulty that IEDs would have in processing said algorithms due to their limited ability to perform computing functions. With these security measures, it is possible to counteract unauthorized modification of data, its manipulation and reproduction, and to avoid MiTM and DoS attacks where, for example, malware could simulate a false IED. To achieve these security requirements in GOOSE and SV messages, IEC62351 has developed two recommendations, IEC62351-6:2007 (currently obsolete) and IEC62351-6:2020. To ensure the authenticity and integrity of the messages, the use of different MAC (Message Authentication Code) algorithms is suggested. Table 1 shows the algorithms proposed by IEC62351-6:2020 to secure the GOOSE and SV protocols [10].

The MAC implementation requires a key pre-shared by both the publisher and the subscribing IEDs. HMAC can provide message authentication using a shared secret instead of using digital signatures with asymmetric cryptography, such as the algorithms proposed by IEC62351:2007. HMAC uses two hash calculation steps. The secret key is first used to derive two keys: internal and external. The first step of the algorithm produces an internal hash derived from the message and the internal key. The second pass produces the final HMAC code derived from the result of the internal hash and the foreign key. Therefore, the algorithm provides better immunity against length extension attacks, so the security of HMAC is guaranteed by the secure distribution of the symmetric key and its unpredictability [24].

Table 2 shows the computational times required by the IEDs and the end-to-end communication time of the different MAC algorithms proposed in IEC62351-6:2020 for the exchange of GOOSE messages with their secure extended frame. As can be seen in Table 2, the end-to-end message exchange, in any of the proposed algorithms, meets the requirement of the IEC61850-5 standard that says that GOOSE messages cannot take more than 3 ms. Therefore, the proposed MAC algorithms to ensure integrity and authenticity can be successfully applied for both GOOSE and SV messaging.

Table 1. MAC algorithms proposed by IEC62351-6:2020.

MAC Algorithm	Hash Function	MAC Value Size in Bytes
HMAC-SHA256-80	SHA256	10
HMAC-SHA256-256	SHA256	16
HMAC-SHA256-256	SHA256	32
AES-GMAC-64		8
AES-GMAC-128		16

Table 2. Times in secure GOOSE messaging exchanges under the different MAC algorithms recommended by IEC62351-6:2020.

Algorithm	Total GOOSE Message Size (bytes)	Publisher Computing Time (ms)	Subscriber Computing Time (ms)	Communication Delay (ms)	End-to-End Delay (ms)
Without security	159	0	0	0.0664	0.0664
HMAC-SHA256-80	193	0.0127	0.0141	0.0709	0.0977
HMAC-SHA256-128	199	0.0127	0.0142	0.0722	0.0991
HMAC256-256	215	0.0127	0.0143	0.0757	0.1027
AES-GMAC-64	205	0.0054	0.0066	0.0730	0.0850
AES-GMAC-128	213	0.0055	0.0069	0.0749	0.0873

4. Digital Security Methodologies outside Those Proposed by IEC62351

ChaCha20 is a stream cipher used to ensure confidentiality, while Poly1305 is a MAC algorithm to ensure the integrity and authenticity of messages. Both algorithms can be used together to form an AEAD (Authenticated Encryption with Associated Data), which verifies both the integrity and authenticity of the ciphertext, as well as the integrity of some extra information.

ChaCha20 is an ARX (add, rotate, mixing columns) of the matrix or smart division of polynomials. AES relies on the fact that modern CPUs have Galois field arithmetic built in as real instructions, while ChaCha20 does not need any of that, which makes it especially useful for low-end electronic devices or those that need great speed [25].

On the other hand, the Poly1305 algorithm is used for the integrity and authenticity of the message. Poly1305 takes a 256-bit key, a message of arbitrary length, and generates a 128-bit (16-byte) output known as a TAG or MAC.

ChaCha20 and Poly1305 are defined in RFC8439 [26]. The combination of both protocols is part of the new TLS 1.3 version, one of the strongest competitors to AES-GCM [27].

RFC8439 provides the series of steps to follow the Poly1305 algorithm, which is detailed in Algorithm 1.

As mentioned above, it is possible to build an AEAD algorithm using ChaCha20 and Poly1305 together, which is why AEAD was given greater emphasis than the use of ChaCha20 and Poly1305 separately, as will be seen below. The algorithm, to perform the Poly1305 MAC TAG, uses the additional text to be authenticated (AAD) and the ciphertext. The AEAD algorithm encrypts the APDU using ChaCha20, and then this text and the additional text are used to calculate the MAC value using Poly1305. To calculate the Poly1305 MAC, a key is generated using a ChaCha20 block with the 32-bit counter block being zero. This MAC value is appended to the "extension" field of the secure GOOSE frame. The length of the "extension" field is reflected in the second byte of the Reserved1 field. The order in which to perform this is detailed in Algorithm 2. Figure 4 shows the process graphically.

Algorithm 1 Poly1305 algorithm

1. Split the key in two.
2. Perform the "clamp" process to the "r" part of the 256-bit key.
3. Set the prime number "P" to (2^{130}-5).
4. Set the "accumulator" variable (Acc) to zero.
5. Divide the message into 16-byte blocks, the last block will most likely be shorter.
6. Read the block as a number in little-endian order.
7. Add a bit after the number of octets, for 16-byte blocks, this would be adding 2^{128}, for the smallest block, adding some power divisible by 8, for example, 2^{120}, 2^{112} up to 2^{8}.
8. Fill the last block with zeros, this has no effect since the block is treated as numbers.
9. Add this number to the accumulator.
10. Multiply by "r".
11. Apply modulo P to the result.
12. Seen mathematically: $Acc = ((Acc + block)r)\%\ P$.
13. Finally, after finishing all the blocks, the value "s" of the key is added and is attached to the accumulator, and the 128 least significant bits are serialized in little-endian order, thus forming the TAG or MAC of the message.

Algorithm 2 AEAD Algorithm applied to the GOOSE APDU

1. Data ← GOOSE-APDU
2. k ← 256-bit pre-shared key
3. E_d ← Encrypt$_k$(Data)
4. k1 ← 256 bits Key for Poly1305
5. h ← MAC$_{k1}$(AAD | padding1|E_d|padding2)
6. GOOSE.Extension ← h
7. GOOSE.APDU ← E_d

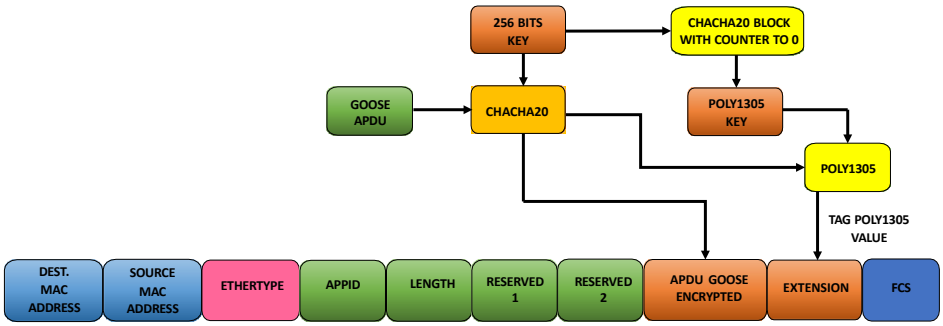

Figure 4. AEAD ChaCha20–Poly1305 process applied to the GOOSE APDU.

In order to decrypt the APDU, the IED breaker, according to the AEAD ChaCha20/Poly1305 algorithm, must first check the integrity of the Poly1305 MAC TAG, if the calculated value matches the MAC attached in the extension field, the GOOSE frame is authentic, and therefore, is decrypted. The series of steps are as follows (Algorithm 3).

Figure 5 shows the decryption process graphically.

The R-GOOSE frame has different fields than the GOOSE frame, according to IEC61850-90-5, so the following fields have been taken to carry out the AEAD ChaCha20/Poly1305 algorithm (Figure 6). Figure 7 shows the authenticity and integrity verification process of the R GOOSE frame using Poly1305, and if the frame is authentic, the session PDU field of the R-GOOSE frame is decrypted using ChaCha20.

Algorithm 3 Algorithm for Poly1305 MAC Verification and ChaCha20 Decryption

1. Data Received ← GOOSE.APDU Encrypted = E_d
2. h ← GOOSE.Extension
3. k1 ← 256 bits Key for Poly1305
4. h1 ← MAC_{k1}(AAD | padding1|E_d|padding2)
5. if h1 = h then
6. k ← Pre-shared key 256 bits
7. APDU ← $Decrypt_k$(Data Received)
8. else
9. return "Invalid TAG, abort decryption process"

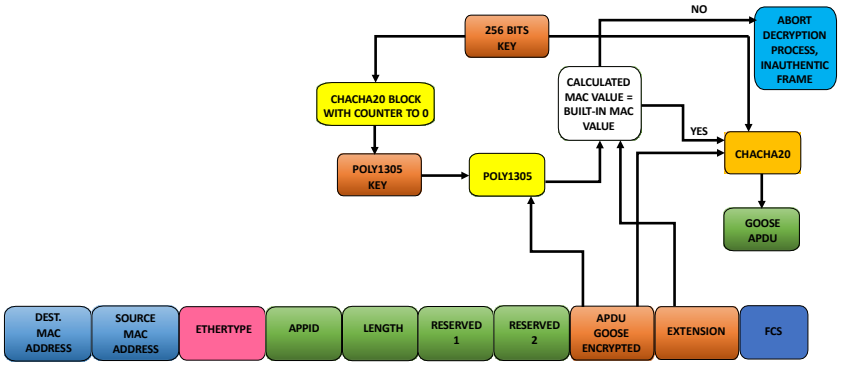

Figure 5. Poly1305 MAC verification and ChaCha20 decryption.

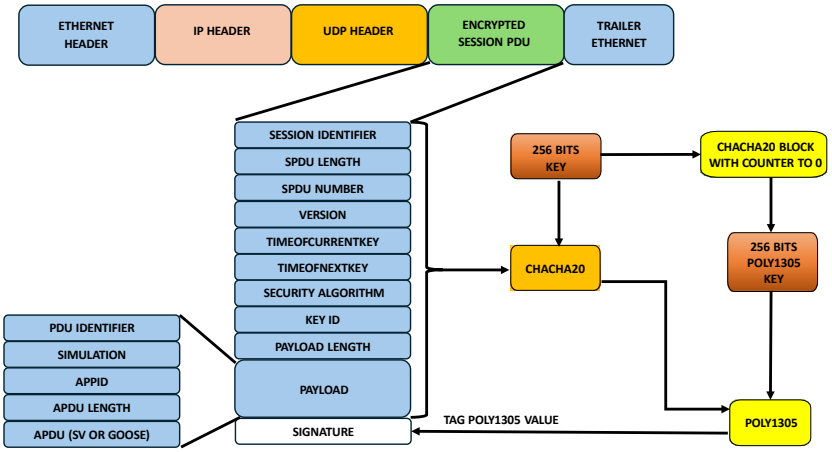

Figure 6. AEAD ChaCha20–Poly1305 process applied to the session PDU field of the R-GOOSE frame.

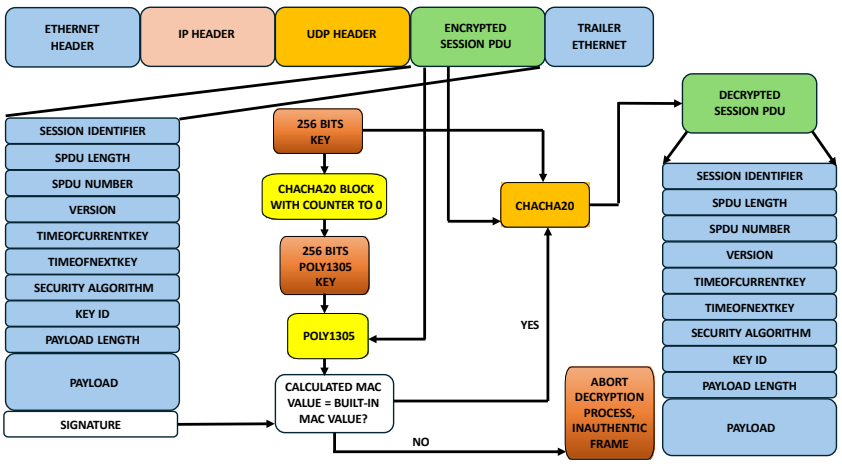

Figure 7. TAG AEAD Poly1305 verification process and decryption applied to R-GOOSE frame.

4.1. Security for GOOSE and R-GOOSE via ChaCha20 and Poly1305, Proposed Method

Although IEC62351-1 specifies that GOOSE and SV algorithms should not use encryption algorithms, the rise of substation digitization and the use of GOOSE and SV for other power applications, such as electromobility, makes it increasingly necessary to have robust encryption algorithms to prevent third parties from obtaining relevant information and at the same time, these algorithms do not affect communication performance.

To send GOOSE frames, the first scenario was carried out, which consists of two laptops that emulate the behavior of a protection IED and a breaker IED, connected through a switch, therefore, both computers are on the same LAN. The structured cabling used is category 6. This can be seen in Figure 8.

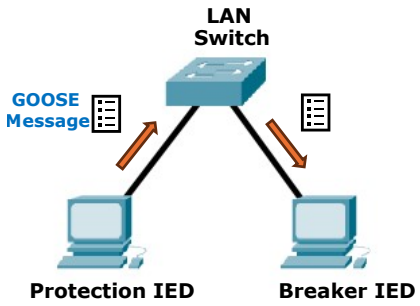

Figure 8. Test scenario for sending GOOSE frames.

In the case of sending R-GOOSE frames, because these must be sent between IEDs that are on different networks, the LAN switch was changed to a router, maintaining the laptops and the structured cabling from the previous scenario. This can be seen in Figure 9.

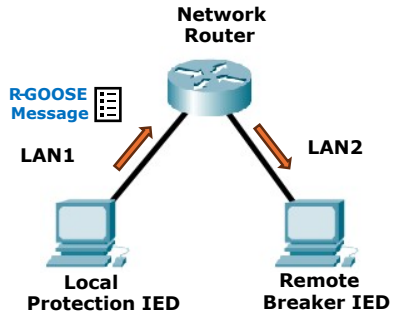

Figure 9. Test scenario for sending R-GOOSE frames.

The technical specifications of the scenarios were as follows:

- Virtual machine on laptops with AMD Ryzen 5 (5600X) processor, with a single vCPU running at 3.7 GHz, 512 MB of RAM with Ubuntu 22 Linux distribution.
- OSI model data link layer switch (2) with eight GigabitEthernet ports (10/100/1000 Mbps).
- OSI model network layer router (3) with two GigabitEthernet ports (10/100/1000 Mbps).
- Structured cabling category 6.

The languages, software, and codes used to conduct the tests were as follows:

- The C programming language was leveraged to implement algorithms enabling encryption with ChaCha20, MAC generation using Poly1305, and in the development of an AEAD algorithm that integrated both functionalities.
- To examine the calculation time of the algorithms, the Python time library was used.
- To examine the GOOSE frame fortified with ChaCha20 and Poly1305 as digital security measures, the S-GoSV software was employed. This software utilizes network interface libraries to meticulously construct and transmit the GOOSE message across the network, ensuring all requisite fields are adequately included [5].

- The Wireshark software was used to visualize the reception of GOOSE and R-GOOSE frames by the laptop emulating the local and remote behavior of the IED broker.
- To carry out the encryption and creation of the Poly1305 MAC TAG for GOOSE and R-GOOSE, the codes from the repository [28] were used, since their operation was proven by testing the examples of RFC 8439.

All the tests carried out are listed below, both for the first scenario (sending and receiving GOOSE frames) and for the second scenario (sending and receiving R-GOOSE frames):

1. Calculation of generation times and transmission delay over the network with digital security measures (Poly1305 versus HMAC-SHA256).
2. Calculation of encryption and decryption times to guarantee the confidentiality of the APDU of the GOOSE message. (ChaCha20 versus AES128-GCM).
3. Calculation of generation times, AEAD verification, communication delay, and end-to-end delay with AEAD digital security (ChaCha20/Poly1305 versus AES128/HMAC-SHA256).

Concerning point 3, the R-GOOSE frame could not be contrasted with other research work, as no articles were found with relevant information.

The presented scenarios include simplified equipment that would typically be found at the bay and process levels of the IEC61850 standard [29].

4.2. Experiment Results

To observe the performance of these algorithms, concerning the GOOSE frame, the beginning of the APDU field until its end has been used, and for R-GOOSE from the beginning of the session identifier field to the end of the APDU field has been considered. This is to compare the results with those obtained by the authors of [30,31], who carried out tests with AES and HMAC-SHA algorithms.

Figure 10 shows the GOOSE APDU in plain text and encrypted, the Poly1305 MAC TAG of the 16-byte GOOSE APDU and the calculation times of both algorithms are also seen. The encryption time for the GOOSE APDU was 0.0033 ms and the TAG generation time was 0.0066 ms.

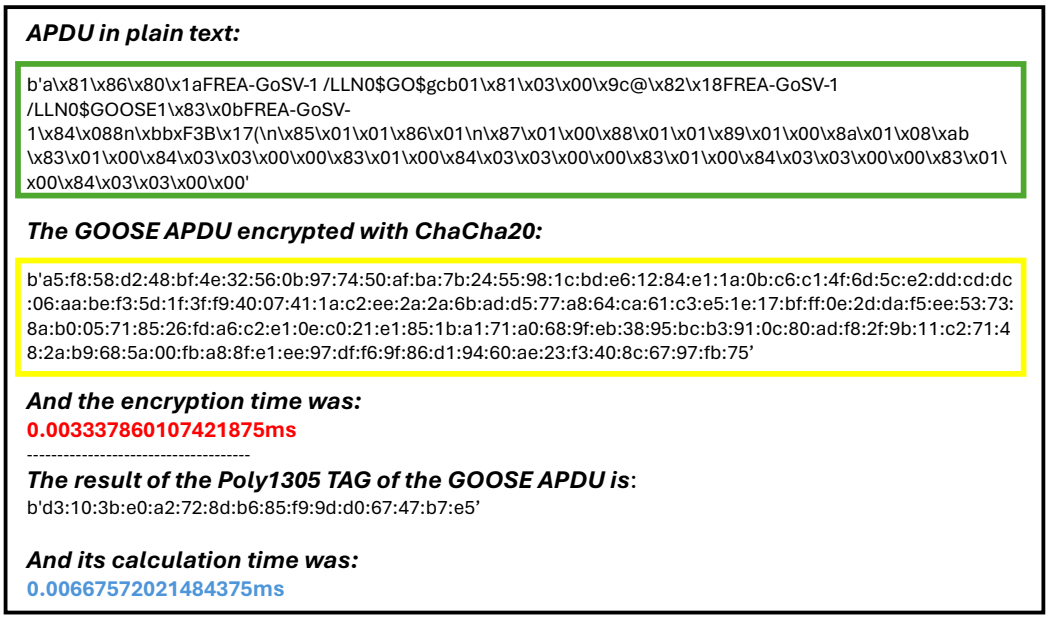

Figure 10. Encryption and TAG of the GOOSE APDU with ChaCha20 and Poly1305.

Figure 11 shows the sending of the GOOSE frame only with authenticity and integrity security measures to be within the stipulations of IEC62351.

Sending time in ms = 0.069000

Sending GOOSE message with digital security values:

01:0c:cd:01:03:ff:08:00:27:ba:d3:ed:88:b8:00:01:00:91:00:26:f2:32:61:81:86:80:1a:46:52:45:41:2d:47:6f:53:56:2
d:31:20:2f:4c:4c:4e:30:24:47:4f:24:67:63:62:30:31:81:03:00:9c:40:82:18:46:52:45:41:2d:47:6f:53:56:2d:31:20:2f:
4c:4c:4e:30:24:47:4f:4f:53:45:31:83:0b:46:52:45:41:2d:47:6f:53:56:2d:31:84:08:38:6e:bb:f3:42:17:28:0a:85:01:0
1:86:01:0a:87:01:00:88:01:01:89:01:00:8a:01:08:ab:20:83:01:00:84:03:03:00:00:30:28:a4:26:80:01:01:81:04:5b:f
c:f6:b0:82:01:3c:84:04:00:00:00:0c:85:10:d3:10:3b:e0:a2:72:8d:b6:85:f9:9d:d0:67:47:b7:e5:

Figure 11. Sending secure GOOSE frame with MAC Poly1305 in the extension field.

Figure 12, in red, shows the *MAC_Tag* field with a hexadecimal value of 0x85 and the *MAC_length* field with a value of 0x10, so the length of the MAC is 16 bytes. Then, in blue, the MAC generated with Poly1305 is seen. When comparing this frame with that of [5], it is observed that the MAC generated with Poly1305 is smaller. It can be noted that the APDU is in plain text since no measures are being used to ensure confidentiality.

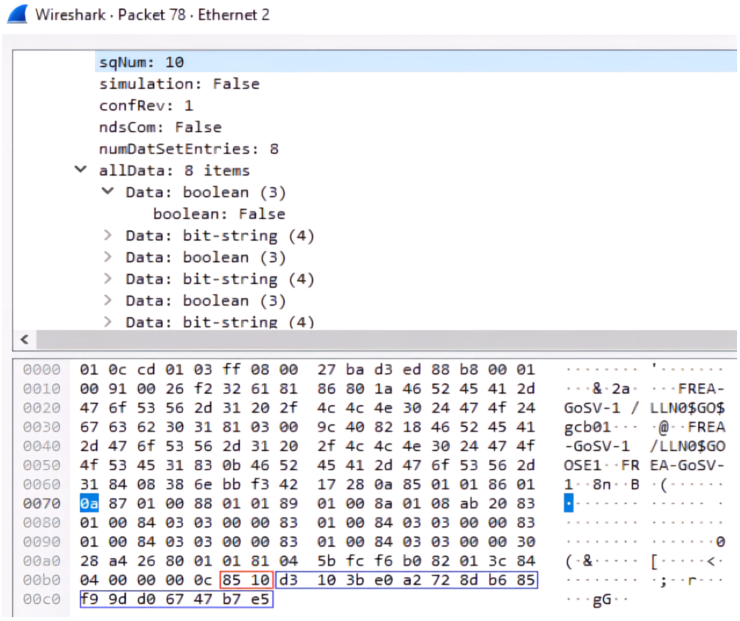

Figure 12. Secure GOOSE frame with Poly1305 received seen via Wireshark.

To observe and compare the calculation times of Poly1305 with HMAC-SHA256, the calculation of HMAC-SHA256 is performed with the same virtual machine that generated the Poly1305 MAC and the calculation time of the authors of [31] is also added. Figure 13 shows the APDU MAC calculated with HMAC-SHA256. It can see that the calculation time of the algorithm is 0.018 ms. Table 3 displays the timings obtained solely through digital security measures intended for verifying the authenticity and integrity of GOOSE messages. It includes a comparison between the results obtained using HMAC-SHA256 in this study and those reported in [31] (third row of the table).

APDU in plain text:

618186801a4745446576696365463635302f4c4c4e3024474f24676362303181030008c4082184745446576696365463635302f4c4c4e3024474f4f534531830b463635305f474f4f5345318408386ebbf34217280a85010186010a870100880101890100 8a0108ab208301008403030000830100840303000083010084030300008301008403030000

HMAC:

512056228cb6497cdcf3aa23a35eb6bd008ec30c8d2372e3d5f0d686369ad7f1

HMAC generation time:
0.018000ms

Figure 13. HMAC-SHA256 algorithm and calculation time applied to the GOOSE APDU.

Table 3. MAC Poly1305 and HMAC-SHA256 computational times.

MAC Algorithm	MAC Size (bytes)	Reference	MAC Generation Time (ms)	Network Sending Delay (ms)
Poly1305	16	Virtual machine, 1 vCPU AMD Ryzen 5 5600X, 512 MB RAM	0.0066	0.069
HMAC-SHA256-256	32	Virtual machine, 1 vCPU AMD Ryzen 5 5600X, 512 MB RAM	0.018	0.1
HMAC-SHA256-256	32	Intel Celeron, 4 GB RAM	0.0127	0.0757

Subsequently, the GOOSE frame is tested with security measures for authenticity, but now also with respect to confidentiality. Figure 14 shows the sending of the GOOSE frame with assurance of authenticity and confidentiality, encrypting the GOOSE APDU and generating its MAC (APDU in plain text) through ChaCha20 and Poly1305.

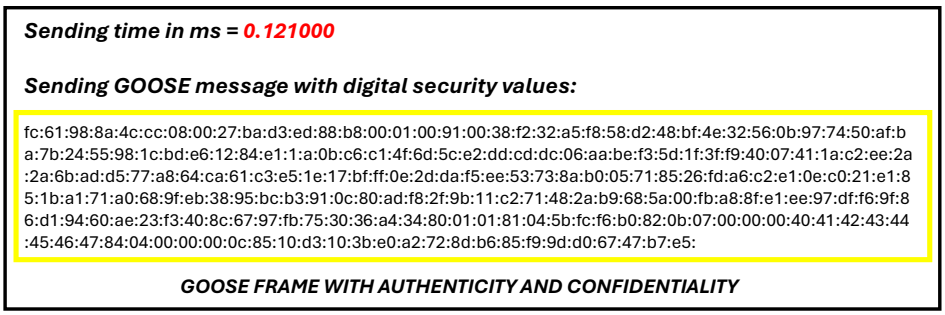

Figure 14. Sending secure GOOSE frame, APDU encrypted with ChaCha20 and Poly1305 MAC embedded in the extension field.

Figure 15 shows the GOOSE frame received on the network with its APDU encrypted using ChaCha20 and the Poly1305 MAC TAG of the GOOSE APDU. It can be seen that the content of the APDU is completely unreadable due to the encryption of the APDU, achieving the confidentiality of the GOOSE message. The MAC TAG of the GOOSE APDU can also be seen in a blue box.

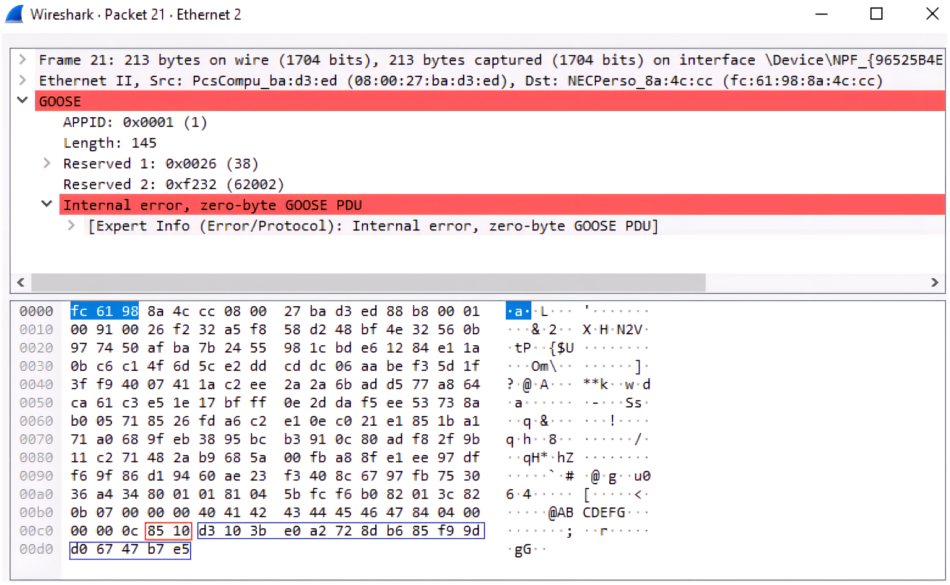

Figure 15. GOOSE frame received with APDU encrypted with ChaCha20 and Poly1305 TAG viewed through Wireshark software.

To compare ChaCha20 with AES128, the time it takes AES128 to perform the encryption is calculated, using the same virtual machine that was used to encrypt the APDU with the ChaCha20 algorithm. Figure 16 shows the result obtained.

ENCRYPTED APDU AES128

8e b0 27 07 6c ee f1 b0 b5 ff ea 9d 4b 02 24 7d 47 59 be 5f ba b3 75 81 25 bb 97 65 da 58 c7 ba d2 c1 ff 1f 8f 2b df 4a 8f 26 4e cd fd e2 7f 0c 11 08 59 86 28 28 99 ac 83 14 4a 9b ae a3 67 4c 30 d7 cd 5a 9b f8 8d d2 e4 df e2 e9 fc 09 49 cf 4e 8a 76 c2 50 e4 ac 83 24 6c c0 2e 6c d4 1d cb b4 b6 d0 8e 20 51 96 da c9 c3 43 6b f9 5b 6b c2 8f c9 4b a3 04 e3 5e 96 43 47 40 b5 36 a8 96 d5 43 3d 89 c1 82 51 71 12 01 e2 cd 7c 0b 5a 21 3f

PACKET DATA SENT:

:01:0c:cd:01:03:ff:08:00:27:ba:d3:ed:88:b8:00:01:00:91:00:38:d5:4b:8e:b0:27:07:6c:ee:f1:b0:b5:ff:ea:9d:4b:02:24:7d:47:59:be:5f:ba:b3:75:81:25:bb:97:65:da:58:c7:ba:d2:c1:ff:1f:8f:2b:df:4a:8f:26:4e:cd:fd:e2:7f:0c:11:08:59:86:28:28:99:ac:83:14:4a:9b:ae:a3:67:4c:30:d7:cd:5a:9b:f8:8d:d2:e4:df:e2:e9:fc:09:49:cf:4e:8a:76:c2:50:e4:ac:83:24:6c:c0:2e:6c:d4:1d:cb:b4:b6:d0:8e:20:51:96:da:c9:c4:43:6b:f9:5b:6b:c2:8f:c9:4b:a3:04:e3:5e:96:43:47:40:b5:36:a8:96:d5:43:3d:89:c1:82:51:71:12:01:e2:cd:7c:0b:5[a]:21:3f:30:36:a4:34:80:01:01:81:04:5b:9c:fd:67:82:01:3c:84:04:00:00:00:01:85:20:f2:e1:71:89:f5:77:d9:51:0c:fc:14:81:57:b0:cf:5f:3c:f0:64:b2:0f:b2:5b:2e:f4:67:41:4d:76:f5:63:78:

ENCRYPTED TIME AES IN ms = 0.017000

Figure 16. Encryption and calculation time of the AES128 algorithm applied to the GOOSE APDU.

Figure 17 shows the APDU decrypted with ChaCha20, to decrypt the APDU the same ChaCha20 algorithm is performed but instead of performing XOR with the plaintext, it is performed with the ciphertext.

The GOOSE APDU decrypted with ChaCha20:

b'61:81:86:80:1a:46:52:45:41:2d:47:6f:53:56:2d:31:20:2f:4c:4c:4e:30:24:47:4f:24:67:63:62:30:31:81:03:00:9c:40:82:18:46:52:45:41:2d:47:6f:53:56:2d:31:20:2f:4c:4c:4e:30:24:47:4f:4f:53:45:31:83:0b:46:52:45:41:2d:47:6f:53:56:2d:31:84:08:38:6e:bb:f3:42:17:28:0a:85:01:01:86:01:0a:87:01:00:88:01:01:89:01:00:8a:01:08:ab:20:83:01:00:84:03:03:00:00:83:01:00:84:03:03:00:00:83:01:00:84:03:03:00:00:83:01:00:84:03:03:00:00'

And the decrypted time was:
0.008821487426757812ms

Figure 17. GOOSE APDU decrypted with ChaCha20 and calculation time.

Table 4 shows the calculation times of each algorithm to guarantee the confidentiality of the APDU of the GOOSE message and it compares it with what was obtained in [31] (third row of the table). The sending of the GOOSE message is encrypted and also authenticated with a MAC algorithm embedded in the extension field.

Table 4. GOOSE APDU encryption and decryption time.

Encryption Algorithm	Reference	Encryption Time (ms)	Decryption Time (ms)	Network Sending Delay (ms)
ChaCha20	Virtual machine, 1 vCPU AMD Ryzen 5 5600X, 512 MB RAM	0.0033	0.0088	0.12
AES128-GCM	Virtual machine, 1 vCPU AMD Ryzen 5 5600X, 512 MB RAM	0.017	0.021	0.10
AES128-GCM	Intel Celeron, 4 GB RAM	0.00797	0.0932	0.079

With respect to R-GOOSE, there are "security algorithm" and "encryption algorithm" fields that identify the algorithms that are being used to provide digital security. To use the ChaCha20 and Poly1305 algorithms, the two bytes of the "security algorithm" field, "encryption algorithm" could be specified to a value of 0X04 and "MAC algorithm" to 0X06 to specify ChaCha20 and Poly1305, respectively. To use the ChaCha20/Poly1305 algorithm it could specify "encryption algorithm" and "MAC algorithm" to 0X05 and 0X06. Table 5 shows what was just stated.

Table 5. Security algorithm and MAC algorithm octets of R-GOOSE frames.

First Octet Value	Encryption Algorithm	Second Octet Value	MAC Algorithm
0	None	0	None
1	AES-128-GCM	1	HMAC-SHA256-80
2	AES-256-GCM	2	HMAC-SHA256-128
3	AEAD AES	3	HMAC-SHA256-256
4	ChaCha20	4	AES-GMAC-64
5	AEAD ChaCha20	5	AES-GMAC-128
		6	Poly1305

The images of the analysis carried out with S-GosV and Wireshark for the R-GOOSE frames can be obtained from the repository available at [32].

Table 6 presents the generation times of the MAC algorithms applied to the R-GOOSE frame and it compares it with what was obtained in [31] (third row of the table).

Table 6. Calculation times MAC algorithms applied from session identifier until the end of the R-GOOSE APDU.

MAC Algorithm	MAC Size (bytes)	Reference	Generation Time (ms)
Poly1305	16	Virtual machine, 1 vCPU AMD Ryzen 5 5600X, 512 MB RAM	0.0083
HMAC-SHA256-256	32	Virtual machine, 1 vCPU AMD Ryzen 5 5600X, 512 MB RAM	0.035
HMAC-SHA256-256	32	Intel Celeron, 4 GB RAM	0.008

Table 7 shows the performance comparison of R-GOOSE message encryption with ChaCha20 and AES256-GCM. Also, it shows the delay in sending the R-GOOSE frame with embedded security measures. It compares it with the result obtained in [31] (third row of the table).

Table 7. Encryption performance from the session identifier field to the end of the R-GOOSE APDU.

Encryption Algorithm	Reference	R-GOOSE Encryption Time (ms)	R-GOOSE Decryption Time (ms)	Network Sending Delay (ms)
ChaCha20	Virtual machine, 1 vCPU AMD Ryzen 5 5600X, 512 MB RAM	0.0030	0.0035	0.117
AES256-GCM	Virtual machine, 1 vCPU AMD Ryzen 5 5600X, 512 MB RAM	0.054	0.002	0.114
AES256-GCM	Intel Celeron, 4 GB RAM	0.286	0.221	[-]

Although the encryption and decryption times of AES-256-GCM are very good, ChaCha20 shows low and very stable values.

The images of the analysis carried out with S-GosV and Wireshark for the AEAD algorithm applied to GOOSE and R-GOOSE frames can be obtained from the repository available at [32].

Table 8 shows the generation and verification times of the AEAD algorithm of the GOOSE message plus the communication delay time and the total delay that considers the processing of the algorithms and the communication delay. In the third row, the times obtained in [30] are shown.

Finally, Table 9 shows a summary of the generation and verification time of the AEAD algorithm applied in the R-GOOSE message.

Table 8. GOOSE frame with AEAD digital security.

AEAD Algorithm	Frame Size (bytes)	Reference	AEAD Generation Time (ms)	Verification Time (ms)	Communication Delay (ms)	End-to-End Delay (ms)
ChaCha20 Poly1305	213	Virtual machine, 1 vCPU AMD Ryzen 5 5600X, 512 MB RAM	0.0145	0.016	0.058	0.0885
AES128 HMAC-SHA256	222	Virtual machine, 1 vCPU AMD Ryzen 5 5600X, 512 MB RAM	0.114	0.121	0.060	0.295
AES128 HMAC-SHA256	222	Intel Celeron, 4 GB RAM	0.0909	0.1047	0.0779	0.2735

Table 9. R-GOOSE frame with AEAD digital security.

AEAD Algorithm	Frame Size (bytes)	Reference	AEAD Generation Time (ms)	Verification Time (ms)	Communication Delay (ms)	End-to-End Delay (ms)
ChaCha20 Poly1305	239	Virtual machine, 1 vCPU AMD Ryzen 5 5600X, 512 MB RAM	0.0147	0.0186	0.11	0.1427

5. Discussion

Cybersecurity in the industrial world is a branch that must constantly evolve, since at the same time cyber attackers propose new and sophisticated hacking tools against public infrastructure. It was understandable that cybersecurity was minimal in these industrial electrical environments, which seemed far away and difficult to attack. However, with the appearance of the Stuxnet, BlackEnergy and CrashOverride malware, it has become clear that these cyber attacks are a reality. And even more so in the IEC61850 standard, since by following communication protocols in a standardized way to achieve interoperability, the possibilities of attacks are increased if it is understood how the semantics of these protocols work. It can be seen that changes in the messages can lead to undesirable openings of breakers.

Cryptography adapts very well to the IEC61850 standard since, thanks to the different cryptographic algorithms, it is easy to notice a change, even if it is minor, in the messages sent on the network. The use of both symmetric and asymmetric keys ensures that there is no one in the middle capable of modifying the messages. These digital security methodologies are becoming more relevant every day since IEC61850 is growing rapidly from the automation of substations towards the complete domain of the administration and control of electrical power systems, where some sensitive data that requires privacy is sent through the GOOSE protocol.

With respect to what IEC62351 and IEC61850-90-5 propose for the digital security of the GOOSE, SV, R-GOOSE, and R-SV protocols, AES algorithms for confidentiality and HMAC or GMAC for integrity and authenticity of these communication protocols do not affect the general performance of the communication and their times adapt to the temporal requirements of maximum 3 ms for these protocols, so confidentiality should be mandatory as is integrity and authenticity in GOOSE and SV according to IEC62351. Also, it has become evident that asymmetric key cryptographic algorithms, such as digital signatures, do not meet these temporal requirements, so they cannot be used. Therefore, using only symmetrical key algorithms is the option that best suits substation communication under the IEC61850 standard.

Regarding the comparison of the AES algorithm in its various formats concerning the ChaCha20/Poly1305 algorithms in their different configurations, the following comments can be made:

- AES is suitable encryption for most modern devices, especially those with Intel processors, which have AES hardware support, making AES operations efficient.
- AES is not ideal for older devices, as it would require software implementations that can be expensive, especially in AES-GCM. This type of encryption is particularly expensive when implemented in software.
- ChaCha20–Poly1305 has been implemented for mobile devices by Google because it is a fast and secure stream cipher, adding it to TLS1.3. This means that if someone finds a flaw or vulnerability in AES-based cipher suites at some point in the future, it will give them a secure and fast option to fall back on. Therefore, the ChaCha20 stream cipher can come in to fill this situation [27,33].
- AES has been more analyzed cryptographically due to its greater age, while ChaCha20 has a solid foundation available in an RFC standard it has been the subject of less cryptoanalysis since its creation.
- ChaCha20 offers a similar level of security to AES. The software implementations of AES can be vulnerable to cache timing attacks, although this is less relevant due to the wide availability of hardware support. For post-quantum security, a 256-bit key is generally recommended, as 128-bit keys can leave open the possibility of batch or multi-target attacks, where multiple users on a system are attacked simultaneously [34,35].

6. Conclusions

The integrity of messages transmitted through GOOSE and R-GOOSE frames is critical for the proper operation of smart grids. Therefore, it is imperative to protect both the integrity and confidentiality of this information. To achieve this, the IEC62351 standard is followed, and AEAD algorithms are considered the most appropriate solution. These algorithms add an extra layer of security while complying with the IEC61850 standard. In conclusion, both standards, IEC62351 and IEC61850-90-5, only propose the use of AES and HMAC algorithms. The assumption that weaknesses are found in these algorithms seriously affects the digital security of IEC61850 communication protocols, leaving them once again exposed to cyber attacks. Having backup cryptographic algorithms is important in continuing to guarantee the digital security of substations under the IEC61850 standard. The study of the ChaCha20 and Poly1305 algorithms was carried out as they are possible substitutes for the AES and HMAC algorithms, with the conclusion that both of these algorithms can satisfactorily adapt to the strict maximum latency requirements that these protocols must sustain.

In all cases analyzed for both the GOOSE and R-GOOSE frames, it can be concluded that the ChaCha20 and Poly1305 algorithms, either operating individually or as an AEAD algorithm, met the strict time requirements (less than 3 ms) set by the IEC61850 standard. It is worth noting that in several cases, the times obtained were equal to or even shorter than those obtained using the AES algorithm and its variants. It is concluded that the presented algorithms are effective due to their simplicity when compared to standard algorithms.

Based on the latest test (Table 9), it was verified that the AEAD algorithm (ChaCha20/Poly1305) meets the requirements of the IEC61850 standard for R-GOOSE frames. The total time for generation, verification, communication, and network delay was less than 1 ms, which is a significant result. This verification has not been previously reported for R-GOOSE frames with the AES AEAD algorithm.

Of the tests carried out, the AEAD algorithm proposed by RFC 8439 is the most notable. This algorithm guarantees the confidentiality, integrity, and authenticity of the GOOSE and R-GOOSE communication protocols within the specified requirements. The performance on the computer that subscribes to these messages with AEAD ChaCha20/Poly1305 digital security is optimal because it first verifies the Poly1305 MAC TAG before decrypting. If the MAC does not match, the frame is immediately discarded without decryption.

S-GoSV is a valuable tool for laboratory testing to observe GOOSE, SV, R-GOOSE, and R-SV frames with various digital security measures. Future work could combine the

ChaCha20 and Poly1305 algorithms to send SV and R-SV frames with digital security, allowing for real-time observation of changes.

Author Contributions: Conceptualization, G.F. and S.D.-C.; methodology, G.F.; software, F.A. and B.S.; validation, B.S.; investigation, F.A. and B.S.; formal analysis, F.A. and B.S.; resources, S.D.-C.; writing—original draft preparation, F.A. and B.S.; review and editing, F.A. and B.S.; supervision, G.F.; project administration, S.D.-C.; funding acquisition, S.D.-C. All authors have read and agreed to the published version of the manuscript

Funding: This research was funded and APC by Ministry of Science and Innovation of Spain under Project PID2022-137680OB-C32.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Gungor, V.C.; Sahin, D.; Kocak, T.; Ergut, S.; Buccella, C.; Cecati, C.; Hancke, G.P. Smart Grid Technologies: Communication Technologies and Standards. *IEEE Trans. Ind. Inform.* **2011**, *7*, 529–539. [CrossRef]
2. Song, Y.; FitzPatrick, G.; Lee, K.; Zhang, Y. Interoperability Test for IEC 61850-9-2 Standard-based Merging Units, In Proceedings of the 2017 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), Washington, DC, USA, 23–26 April 2017. [CrossRef]
3. Yegorov, P.K.; Lackovitch, A.; Dean, E.; Mustafa, H.M.; Basumallik, S.; Srivastava, A. Analyzing GOOSE Security in IEC61850-based Substation Using ML, SDN and Digital Twin. In Proceedings of the 2023 North American Power Symposium (NAPS), Asheville, NC, USA, 15–17 October 2023; pp. 1–6. [CrossRef]
4. Hussain, S.M.S.; Ustun, T.S.; Nsonga, P.; Ali, I. IEEE 1609 WAVE and IEC 61850 Standard Communication Based Integrated EV Charging Management in Smart Grids. *IEEE Trans. Veh. Technol.* **2018**, *67*, 7690–7697. [CrossRef]
5. Farooq, S.M.; Hussain, S.S.; Ustun, T.S. S-GoSV: Framework for Generating Secure IEC 61850 GOOSE and Sample Value Messages. *Energies* **2019**, *12*, 2536. [CrossRef]
6. Kush, N.; Ahmed, E.; Branagan, M.; Foo, E. Poisoned GOOSE: Exploiting the GOOSE protocol. In Proceedings of the Twelfth Australasian Information Security Conference, AISC'14, Auckland, New Zealand, 20–23 January 2014 ; Volume 149; pp. 17–22.
7. Ahn, H.; Choi, J.; Kim, Y.H. A Mathematical Modeling of Stuxnet-Style Autonomous Vehicle Malware. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 673–683. [CrossRef]
8. UCA International Users Group WG15 Public Site. 2016. Available online: http://iectc57.ucaiug.org/wg15public(accessed on: 2 February 2024.
9. Ustun, T.S.; Farooq, S.M.; Hussain, S.M.S. A Novel Approach for Mitigation of Replay and Masquerade Attacks in Smartgrids Using IEC 61850 Standard. *IEEE Access* **2019**, *7*, 156044–156053. [CrossRef]
10. Hussain, S.M.S.; Ustun, T.S.; Kalam, A. A Review of IEC 62351 Security Mechanisms for IEC 61850 Message Exchanges. *IEEE Trans. Ind. Inform.* **2020**, *16*, 5643–5654. [CrossRef]
11. Hong, J.; Karnati, R.; Ten, C.W.; Lee, S.; Choi, S. Implementation of Secure Sampled Value (SeSV) Messages in Substation Automation System. *IEEE Trans. Power Deliv.* **2022**, *37*, 405–414. [CrossRef]
12. Da Silveira, M.G.; Franco, P.H. IEC 61850 network cybersecurity: Mitigating GOOSE message vulnerabilities. In Proceedings of the 6th Annual PAC World Americas Conference, Raleigh, NC, USA, 20–22 August 2019; pp. 1–9.
13. Firouzi, S.R.; Vanfretti, L.; Ruiz-Alvarez, A.; Hooshyar, H.; Mahmood, F. 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. *Electr. Power Syst. Res.* **2017**, *144*, 255–267. [CrossRef]
14. Hariri, M.E.; Youssef, T.; Harmon, E.; Habib, H.; Mohammed, O. The IEC 61850 Sampled Measured Values Protocol: Analysis, Threat Identification, and Feasibility of Using NN Forecasters to Detect of Spoofed Packets. In Proceedings of the 2019 IEEE International Conference on Environment and Electrical Engineering and 2019 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe), Genova, Italy, 11–14 June 2019; pp. 1–6. [CrossRef]
15. Falk, H.; Adamiak, M.G.; Baigent, D.; Madani, V. An overview of the new IEC 61850 synchrophasor publish-subscribe profile. In Proceedings of the 2013 66th Annual Conference for Protective Relay Engineers, College Station, TX, USA, 8–11 April 2013; pp. 309–321.
16. Kaura, C.; Sindhwani, N.; Chaudhary, A. Analysing the Impact of Cyber-Threat to ICS and SCADA Systems. In Proceedings of the 2022 International Mobile and Embedded Technology Conference (MECON), Noida, India, 10–11 March 2022; pp. 466–470. [CrossRef]
17. Geiger, M.; Bauer, J.; Masuch, M.; Franke, J. An Analysis of Black Energy 3, Crashoverride, and Trisis, Three Malware Approaches Targeting Operational Technology Systems. In Proceedings of the 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vienna, Austria, 8–11 September 2020; Volume 1, pp. 1537–1543. [CrossRef]
18. Slowik, J. Anatomy of an Attack: Detecting and Defeating Crashoverride. 2018. Available online: https://www.virusbulletin.com/uploads/pdf/magazine/2018/VB2018-Slowik.pdf (accessed on 5 March 2024).

19. Hassankashi, M. Security on the Web by Advanced Encryption Standard (AES) and Security Assertion Markup Language (SAML)—codeproject.com. Available online: https://www.codeproject.com/Articles/1023379/Security-On-The-Web-By-Advanced-Encryption-Standar (accessed on 3 February 2023).
20. Abdullah, A.M. Advanced encryption standard (AES) algorithm to encrypt and decrypt data. *Cryptogr. Netw. Secur.* **2017**, *16*, 11.
21. Vadaviya, D.; Tandel, P. Study of avalanche effect in AES. In Proceedings of the National Conference on Recent Advances in Engineering for Sustainability, Surat, India, 29 May 2015.
22. Gürkaynak, F.K. GALS system design: Side Channel Attack Secure Cryptographic Accelerators. Ph.D. Thesis, ETH Zurich, Zürich, Switzerland, 2006.
23. Park, B.; Song, J.; Seo, S.C. Efficient Implementation of a Crypto Library Using Web Assembly. *Electronics* **2020**, *9*, 1839. [CrossRef]
24. Hussain, S.M.S.; Farooq, S.M.; Ustun, T.S. Analysis and Implementation of Message Authentication Code (MAC) Algorithms for GOOSE Message Security. *IEEE Access* **2019**, *7*, 80980–80984. [CrossRef]
25. Reza, S.M.S.; Ayob, A.; Arifeen, M.; Amin, N.; Md Saad, M.H.; Hussain, A. A lightweight security scheme for advanced metering infrastructures in smart grid. *Bull. Electr. Eng. Inform.* **2020**, *9*, 777–784. [CrossRef]
26. @RustCryptoOrg. chacha20 v0.9.1. Available online: https://crates.io/crates/chacha20 (accessed on 14 January 2024).
27. Nir, Y.; Langley, A. ChaCha20 and Poly1305 for IETF Protocols. RFC 84392018. Available online: https://doi.org/10.17487/RFC8439 (accessed on 20 February 2024).
28. Kueltz, A. GitHub Repository RFC7539 v2.0.1. Available online: https://github.com/AntonKueltz/rfc7539 (accessed on 15 December 2023).
29. Roa, O.; Botero, J.F.; Gutierrez-Betancur, S.A.; Tobar-Rosero, O.A. GOOSEAttacker: Synthetic Attack Generation Tool for IEC61850. In Proceedings of the 2023 IEEE Latin-American Conference on Communications (LATINCOM), Panama City, Panama, 15–17 November 2023; pp. 1–6. [CrossRef]
30. Hussain, S.M.S.; Farooq, S.M.; Ustun, T.S. A Method for Achieving Confidentiality and Integrity in IEC 61850 GOOSE Messages. *IEEE Trans. Power Deliv.* **2020**, *35*, 2565–2567. [CrossRef]
31. Ustun, T.S.; Farooq, S.M.; Hussain, S.M.S. Implementing Secure Routable GOOSE and SV Messages Based on IEC 61850-90-5. *IEEE Access* **2020**, *8*, 26162–26171. [CrossRef]
32. Alonso, F.; Samaniego, B. GitHub Repository. 2024. Available online: https://github.com/Pancho41y/Chacha20-Poly1305 (accessed on 20 February 2024).
33. Sullivan, N. Do the ChaCha: Better Mobile Performance with Cryptography-blog.cloudflare.com. 2015. Available online: https://blog.cloudflare.com/do-the-chacha-better-mobile-performance-with-cryptography/ (accessed on 10 December 2023).
34. Sönmez, B.; Sarıkaya, A.A.; Bahtiyar, S. Machine Learning-Based Side Channel Selection for Time-Driven Cache Attacks on AES. In Proceedings of the 2019 4th International Conference on Computer Science and Engineering (UBMK), Samsun, Turkey, 11–15 September 2019; pp. 1–5. [CrossRef]
35. Mushtaq, M.; Akram, A.; Bhatti, M.K.; Rais, R.N.B.; Lapotre, V.; Gogniat, G. Run-time Detection of Prime + Probe Side-Channel Attack on AES Encryption Algorithm. In Proceedings of the 2018 Global Information Infrastructure and Networking Symposium (GIIS), Thessaloniki, Greece, 23–24 October 2018; pp. 1–5. [CrossRef]