Cape Peninsula
University of Technology

DEVELOPMENT OF AUTHENTICATION ALGORITHMS FOR IEC 61850 GOOSE AND SAMPLED VALUE MESSAGES

by

SIKHO AFRICA NDZEKU

Thesis submitted in fulfilment of the requirements for the degree

Master of Engineering: Electrical Engineering

in the Faculty of Engineering and Built Environment

at the Cape Peninsula University of Technology

Supervisor: Doctor Quinton Bart

Bellville Campus

Date submitted August 2023

DECLARATION

I, Sikho Africa Ndzeku, declare that the contents of this dissertation/thesis represent my unaided work and that the dissertation/thesis has not previously been submitted for academic examination towards any qualification. Furthermore, it represents my own opinions and not necessarily those of the Cape Peninsula University of Technology.

[illegible] 01/11/2023

Signed **Date**

ABSTRACT

Before standardising uniform communication networks, protocols such as the Distributed Network Protocol (DNP3), were used in distributed control systems to transmit telemetry data. However, these network protocols lacked a standard naming convention, making their implementation in control systems expensive due to the use of copper wiring.

The Industrial Internet of Things (IIoT) has contributed to the integration of Information and Communication Technology (ICT) in power systems. The integration of smart technology in substation automation has led to the transformation of the Smart Grid (SG). Over the years, malware and other cyber-attacks have compromised the cyber-security of industrial networks. Some attacks have successfully hacked the Supervisory Control and Data Acquisition (SCADA) systems of industrial plants. Cyber-security is becoming a concern in substation automation and is gaining attention in power systems. Protecting all information in Substation Automation Systems (SAS) is of paramount importance for the success of the SG revolution.
The need to standardise communication networks prompted the transition to networked smart grid systems, reducing costs and engineering time associated with system implementation. The number of security threats targeting electrical networks has been increasing rapidly, and several protocols utilised in these environments are being studied to address these cyber-attacks. The development of security mechanisms for securing the substation communication network is crucial.

In the realm of Cyber-Physical Systems (CPS), the IEC 61850 standard for communication networks is attracting significant attention for its potential to modernise, technologically advance, and make distribution automation effective and economical. IEC 61850 provides an integrated solution in the power system for communication between intelligent devices, offering interoperability and reliability, and incorporating a better form of standardisation as the central smart grid communication protocol. However, IEC 61850 does not have any safety-related features, and cyber-security attacks remain a concern in the substation environment. Edition 1 of IEC

61850 did not emphasise cybersecurity as a primary focus. The standardisation of data models and communication protocols were the main objectives of this version. Furthermore, Edition 2 of IEC 61850 introduced some improvements to the standard. Although Edition 2 incorporated the enhancement of security features compared to Edition 1, it did not offer a comprehensive set of cybersecurity capabilities that addressed all modern security challenges.

The deployment of IEC 62351 has been introduced to address this concern in power systems. It is crucial to secure communication in the SAS from any cyber-security attacks. Implementing both IEC 61850 and IEC 62351 standards requires extensive knowledge in data networking, software modelling, system simulation, and testing procedures.

Generic Object-Oriented Substation Event (GOOSE) and Sampled Value (SV) messages are critical for secure operation and have stringent performance requirements. GOOSE is a model process where data is collected, grouped into a dataset, and transmitted on a process bus. SV is a publisher and subscriber communication where information, such as values of power, is transmitted between the merging units and intelligent electronic devices (IEDs). Compromising GOOSE or SV messages may cause severe power loss to the system. Manipulating or disrupting GOOSE communications could compromise the efficient operation of protective relays, resulting in delayed or incorrect responses under fault events. Tampering with SV messages may result in false data being fed into the protection and control systems, causing the protective devices to malfunction or fail. Both situations have the potential to disrupt the power grid's normal operation, resulting in failures and power outages that have severe consequences for vital systems and end users. As a result, securing these communication protocols is critical to ensuring the reliability and stability of the electrical infrastructure.

This study aims to develop an authentication algorithm for Routable-GOOSE (R-GOOSE) or Routable-SV (R-SV) and implement it in a real-time software application. Critical security features must be enabled to support authentication and authorisation. The EtM algorithm is proposed for maintaining message confidentiality and integrity, with AES-128 encryption for privacy and MAC algorithms for message authentication.

Simulation results indicate that the EtM algorithm can be successfully used for R-SV messages while meeting the stringent 3 ms latency criteria. The results suggest that future IEC 62351 security standards can confidently advocate for encryption for R-SV communication.

ACKNOWLEDGEMENT

I would like to express my gratitude to the following:

- The Almighty God for the provision of life, health, and energy to be able to fulfil my dreams to this level.
- Special thanks to my supervisor, Doctor Bart, for guidance and advice. His patience, wisdom, and advice have helped me a great deal towards completing this thesis.
- To all my friends, both in and out of the department, for your patience and help – a big thank you.
- To my family for their never-ending support and unremitting belief that I could do this.

DEDICATION

This thesis is dedicated to my mother, Nomonde Leticia Ndzeku, and my late fiancé, Azile Nokwindla Holayi. Further dedication goes to the entire family for all their committed support and patience. May God continue to bless us!

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY OF TERMS

- **International Electrotechnical Commission**
  The International Electrotechnical Commission (IEC) prepares and publishes International Standards for all electrical, electronic, and related technologies.

- **IEC-61850**
  The standard for communications at a substation.

- **IEC-62351**
  The international information security standard for operations in the control of power systems.

- **IEC TC57 WG15:**
  IEC 62351 requirements for power system security. The communication standard was initiated to assume the evolution and implementation of cybersecurity standards for substation automation system communications.

- **IED**
  Intelligent Electronic Device. A microprocessor-based system that performs protection, measuring, and control functions in the substation automation industry.

- **RTDS**
  Real-Time Digital Simulator for power systems. It can simulate any current substation system in real-time.

- **GOOSE**
  The Generic Object-Oriented Substation Events protocol is event-based. GOOSE is specified as a publisher/subscriber type communication to distribute information exchange between IEDs across a substation network over Ethernet.

- **SAMPLED VALUE**
  Sampled Values (SV) protocol is a publisher/subscriber type communication. SV is used for information exchange between IEDs in a Substation over the Ethernet. SV is used for sending digitised values of power system quantities.

- **Substation Configuration Language**
  Substation Configuration Language (SCL) reference’s part 6 of the IEC61850 standard. SCL provides a mechanism for defining how substation equipment is connected to the substation.

1. CHAPTER ONE: INTRODUCTION

1.1 Introduction

Power systems have long been a crucial component of infrastructure since they provide dependable and uninterrupted power to people's homes, workplaces, and factories all around the world. Cybersecurity is becoming increasingly crucial, not only for these physical systems but also for the software platforms that operate and maintain them. Cyberattacks are growing more widespread, sophisticated, and destructive. As a result, power systems are now vulnerable to similar attacks. Indeed, multiple cyberattacks on electrical systems have been reported in recent years, with some causing significant damage. Given this risk, both power systems and the software that supports them must be appropriately secured.

Cyber security in power systems is an important component of the total resilience of the electrical grid, as digital attacks on such networks can be disastrous for critical infrastructure. As a result, industry professionals and government institutions must work tirelessly to remain ahead of harmful attack attempts. Many major victories have been achieved in this regard: effective security solutions have been developed; protocols have been upgraded to better protect connected systems from attack; and best practices are periodically examined to assure regulatory compliance.

Researchers and the electrical sector have paid close attention to authentication algorithms for IEC 61850 GOOSE and Sampled Value messages (SVM) in recent years. The reason for the attention: first, these are the two most commonly used communication formats in smart grids and second, many people are concerned about their security and privacy. Unfortunately, if these messages are not properly verified, they can be readily compromised. Several sectors are profiting from the use of IEC 61850 GOOSE and Sampled Value messages, as well as the implementation of IEC 62351 security measures. GOOSE and Sampled Value messages are ethernet-based systems and hence introduce a risk of cyber security. Authenticated communication is essential in many security applications. The number of such applications is growing, but more efficient and safe authentication procedures are still required. We introduce a general architecture in this thesis for creating authentication methods for IEC 61850 GOOSE and Sampled Value messages. We begin by outlining the fundamental structure and elements of an authentication algorithm. The development of our framework for the authentication algorithm is then thoroughly described. Finally, using a set of best practices and actual cases, we assess the suggested framework. Also, we conducted simulation tests to assess the performance and efficiency of our algorithm.

As a result, even though there will always be a chance that power systems' cyber security may be compromised, taking these preventative measures (implementation of IEC 62351 security measures) greatly lowers the potential damage. Organisations must act proactively to safeguard their digital assets given the constantly changing dangers posed by criminal actors. As a result, a thorough strategy that takes into account organisational and operational factors in addition to technological ones must be established. Strong authentication and authorisation mechanisms, encryption of sensitive data in storage and transit, ongoing monitoring of online behaviour, routine system patching, prompt reaction to new threats and vulnerabilities, and awareness training for staff are all required as a result. Companies can successfully reduce risk and improve overall cybersecurity by implementing these actions.

1.2 Overview

For accurate, reliable monitoring and control in the substation, and to ensure quality power in the electrical network and protection (including security mechanism) of the infrastructure in case of faults or cyber-attacks, secure Supervisory Control and Data Acquisition (SCADA) applications were implemented (Ncube, 2012). SCADA operations are critical for data control, data monitoring, and equipment control for operational maintenance. SCADA systems are based on communication and many protocols have been deployed in the system to allow for data exchange. SCADA systems are used to make use of legacy communication protocols such as DNP3, Modbus, etc. These protocols are regarded as non-standardised legacy communication protocols. These protocols were either based on RS-232 or RS-485 which were limited to a data transfer rate of less than 1Mbps, which greatly negatively affects execution time and data access as information can be easily intercepted when the transmission of information data is low (Ncube, 2012). Figure 1.1 shows a solution where non-standardised ethernet-based Intelligent Electronic Devices (IEDs) are implemented. One of the advantages of ethernet-based systems includes the use of less copper wiring (and hence the connection time is less), inexpensive cabling cost, and network addressability.

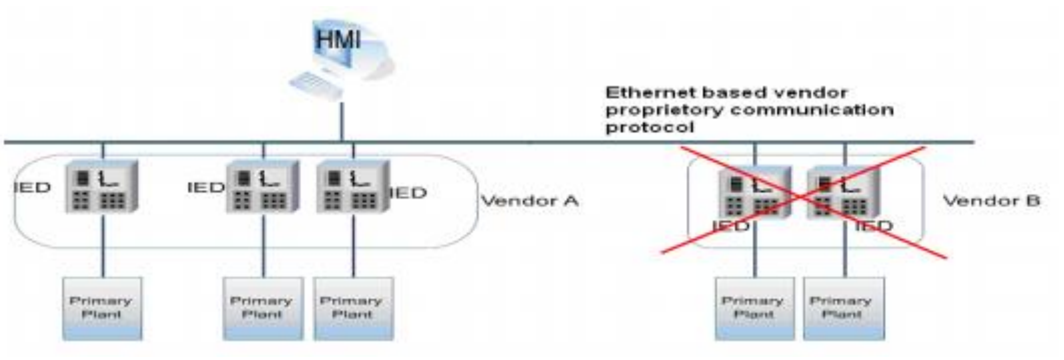


Figure 1.1: Substation architecture using non-standardised IED (Emmanuel, 2014).

Cyber security is essential for the safe and efficient operation of the intelligent electrical network. (Leszczyna, 2018) agrees that standardisation should be applied throughout. Recently, many new standards have been published to address communication networks and cyber security within substation automation. In the 21st century substation automation has become one of the most interesting topics and hence the two standards of IEC 61850 and IEC 62351 in power system networks. Both IEC standards have gained and attracted worldwide attention as the dominant topics in power systems. As such for this research, the focus will be on the following applicable standards, IEC 61850, and IEC 62351 which define communication networks and cyber security requirements for power systems.

At most, the violation of cyber security may result in a business loss in terms of finances and other major consequences as a result. Cyber-attacks on the smart grid may be harmful having a disadvantaged impact on the health, safety, or economic situations of the general population. Securing the smart grid is one of the paramount fundamental procedures that need to be taken to protect the grid from attacks and this requires multidisciplinary approaches and procedures that form a combination with various technologies, policies, and standards. The International Society of Automation (ISA) introduced security standards for the industrial control system. The security requirements are also well suited for the SCADA and Information Technology (IT) environment (Karnati, 2020):

1. Access control,
2. Usage control,
3. Data privacy,
4. Data confidentiality,
5. Limit data flow,
6. Timely event response and,
7. Network resource availability.

IEC 62351 is the standard for cyber security to help secure IEC 61850 communication. (Hohlbaum et al., 2010) however, alludes to some challenges of the standard that need to be addressed. Secure communication in the real-time substation environment must be addressed per the IEC 61850 real-time specification. This can be achieved by complying with the technical specifications of IEC 61850 GOOSE and SV messages as specified in IEC 62351-6.

1.2.1 Cyber Security

Networking applications are increasingly in demand for secure communication. Finding a method to verify data security transmission over an unstable and insecure medium is a necessity. The demand for secure data communication has initiated the development of cryptographic standards and encryption algorithms (Khali et al., 2016). The Hash Message Authenticated Code (HMAC) is considered a preferred standard for authentication with robust security features. It is a signed security tag used to secure plaintext employing authentication (Karnati, 2020). As such, the Secure Hash Algorithm (SHA)-2 hash function was introduced with improved security levels of the Advanced Encryption Standard (AES). HMAC and digital signature algorithms (DSA) rely heavily on hash functions. The first secure hash function algorithm, SHA-1, was released in 1995; and used in numerous network-based applications, however, is being phased out. SHA-1 is used for message authentication with the shared secret key. The implementation of HMAC is to provide authentication to both the source of the message and its integrity achieved by attaching a digital signature to the message (Michail et al., 2004). Therefore, the receiver of the communication is in a position to determine whether or not the message has been altered maliciously by the cyber-attacker. Nonetheless, both the sender and the receiver need to have access to the same secret key to identify any changes made to the message. Due to the increasing demand for internet technology, essential security implementation is required. Furthermore, another discussion on high-performance development in security development and implementation is required.

Daily, the use of the internet is rapidly increasing and as such, it is expected that internet networking is secured. Internet together with cyber security is an essential, critical, and crucial subject in today’s information and cyber systems. The transmission of data or information must be protected against illegal and unauthorised access by providing special security measures. Digital signatures, which may be used to authenticate a communication (encrypted or not), are one of the security techniques employed to protect information by examining the original content of the message that has not been tampered with. One of the advantages of the digital signature is that it cannot be imitated as such, the message is time-stamped and signed. The process includes initiating the identity of the signatory and the integrity of the message can be verified (Alajbegović et al., 2006). As such, integrity and authentication cannot be separated as encryption alone does not provide integrity nor does it provide authentication.

(Oyelade et al., 2015) studies the implementation of secured message transmission using the Data Encryption Standard (DES) and Rivest Shamir-Adleman (RSA) cryptosystem. In the past, cryptosystem was used to secure and protect confidential information in the defence force and important state institutions such as national security. Security methods are continuously being used to secure critical and confidential data against cyber-attacks. As mentioned, essential information needs to be protected against any penetration and tampering; the information must also be transmitted confidentially. (Oyelade et al., 2015) utilise the encryption and decryption mechanism using an asymmetric algorithm to provide a secure channel. Furthermore, the transmitted data will be encrypted and decrypted using the symmetric algorithm. It is critical to maintain the authenticity or security of digital data while it is being transmitted across a network; encryption is a critical component in maintaining data integrity and security (Pedamkar, 2023).

The implementation of the cryptosystem is to address authentication, confidentiality, integrity, and non-repudiation security features. Authentication is a function or process related to the verification and determination of the source of data, verifying a user’s identity. Confidentiality ensures the privacy of the data. Only authorised users can possess the contents of the information. Confidentiality is designed to prevent malicious attacks such as snooping. Data integrity is designed to protect the contents of the information to ensure that it is not altered by authorised users. When sending or receiving data, the non-repudiation service protects against data being retracted by either the sender or the receiver. Non-repudiation is not part of integrity; it is a security feature that prevents an intruder from denying the validity of their previous actions and provides evidence that an action occurred.

(Oyelade et al., 2015) further studies the implication of speed and strength of the symmetric algorithm with the strength and key management capabilities of the asymmetric algorithm. It is noted that the symmetric algorithm provides a poor key management technique and as such the implementation of the algorithms is such that the key generation is processed using the asymmetric encryption technique, RSA algorithm. RSA algorithm technique is known for its advantages in encryption and authentication when employed. The process used for encrypting and decrypting the data is symmetric encryption. Successful transmission of the text document data (using client-server) is concluded using an enhanced encryption algorithm that combines the strengths of both the symmetric and asymmetric key algorithms. The technique was implemented in Microsoft Visual Basic. NET. It was further found that the combination of these two key algorithms provides an increase in the encryption speed. As stated on paper, the technique provides better security when used together.

1.3 Awareness of Problem

The IEC 61850 and IEC 62351 standards are being implemented globally in the substation automation engineering field for protection, monitoring, and control. The development and implementation of the standards are well documented in scientific papers. Examples of typical applications can be found in (Farooq et al., 2019), (Schlegel et al., 2017a), (Hohlbaum et al., 2010) and (Pal & Dash, 2015). The efficient operation of the power network requires secure, failsafe, accurate, and reliable mechanisms to ensure that network integrity is maintained for optimal operation. The implication is that any tool or mechanism utilised to assist with the reliable operation of the power system should inherently be robust, accurate and effective. Viruses and cyber-attacks have compromised the cyber-security of industrial networks. Delivery of messages is delayed as a result of these attacks (Denial-of-Service, Man-in-the-Middle etc.), and data loss and data security become increasingly important when interacting with network components. Cybersecurity is a growing concern in substation automation, and it is important to protect all information in Substation Automation Systems (SAS). However, IEC 61850 does not have any safety-related features, and the deployment of IEC 62351 is necessary to address cyber security concerns.

The smart grid and substation automation networks are installed with IEDs which operate on communication protocols that need to conform to IEC 61850 and the implementation of the IEC 62351 security standard is also to be prioritised. Information infrastructure that enables power system protection, monitoring, and control is essential to substation automation. Previously, communication networks were not vulnerable to security measures as communication occurred via private networks which implies that they were secured through Security Through Obscurity (STO). This meant that a system can be secure so long as nobody outside of its implementation group can find out anything about its internal mechanisms. The research into the security of communication networks in substation automation has driven the need for standardisation of power system communication protocol as security and integrity became a never-lasting concern.

With the increasing installation of IEDs in the power system network, it is essential to implement secure communication for substation equipment. Problems in cyber security can arise when communication is not secured or unreliable. The power system needs to be protected against cybersecurity attacks and hence security for IEDs is critical. Security attacks may come in the form of deliberate attacks launched by disgruntled employees or hackers who can compromise the substation network. One of the ways to prevent intrusion of systems is to install the Intrusion Detection System (IDS) and Intrusion Prevention System (IPS). IDS protects SCADA and substation automation systems against cyber-attacks. IDS detect uncertified and harmful actions on the substation network and as well identifies problems in the communication, allowing it to detect malfunctions in the substation. Its unique approach automatically creates a complete system model of the automation system and the substation. An intrusion prevention system (IPS) is software that performs all of the functions of an IDS while additionally attempting to prevent cyberattacks. It not only detects malicious activity, but it also takes steps to prevent attacks from any suspicious activity. The IPS may drop a packet from suspicious traffic, automatically terminate a port, or block future network traffic from that IP address.

Security standards need to be assessed to address vulnerabilities in the smart grid infrastructure. To protect the Confidentiality, Integrity, and Availability (CIA) of substation networks, the communication in power systems needs to be safe from cyberattacks. Confidentiality can be managed by encrypting data and files and requiring multi-factor authentication. Integrity can be achieved by using cryptography and digital signatures to securely check and prove the integrity of data. Availability monitoring and maintaining hardware and software. Security mechanisms in the power system network must be implemented by achieving a balance in information technology and power system operations. This study will contribute towards the adoption of the IEC 62351 standard.

1.4 Problem Statement

1.4.1 Ideal Scenario

An accurate security mechanism will ensure that the power system will be efficient, robust, reliable, and protected from cyber-attacks. One of the most recommended practice steps that can be considered in a power system environment is to conduct an industrial control system security risk assessment. However, that might not be enough; the authentication and encryption security control form the basis for most cyber security solutions techniques utilised in power systems.

The cyber security strategy must protect the assets most critical to an efficient and reliable power system operation. All communication networks to the power system must be secured and continually assessed for any changes and new vulnerabilities. Defensive strategies need to be put in place to detect when a potential vulnerability is exploited. Critical security controls can reduce the risk of cyber-attacks by great margins.

The industrial environment is to implement and develop a strong security control mechanism to increase operational efficiency and reduce the risk of attacks through the standardisation of IEC 61850 and IEC 62351. What typifies a secure power system and hence reliable and efficient is that implements authentication or encryption security controls conforming to communication and cyber security standards.

1.4.2 The Problem Statement

There is a need to develop robust authentication algorithms for IEC 61850 GOOSE and SV messages in modern networked substations to prevent cyberattacks and improve the security of these systems. Within this standard, a low latency restriction must be met, as well as time-critical status transfer. Hence a need for high-speed time-critical communications for updating the status and events in the power system. The development of the thesis will include the design and implementation phase. The design phase will focus on the code development of the authentication algorithm adhering to the correct frame structure for GOOSE/SV messages. The implementation phase is the commissioning of the presented authentication algorithm using the Kali Linux command line to run the code, Wireshark to collect and analyse data, and Ettercap to launch man-in-the-middle attacks (MITM) to infect traffic between the publisher-subscriber devices.

There are many substations across the world and each one of them is critical for achieving the efficiency and adaptability of the smart grid network. As part of the operation of the smart grid, information about the consumption and operations is required to be transmitted to a substation automation system for analysis. This however requires data to be transmitted via two-way communication; this is achieved by Ethernet and TCP/IP and conforms to the IEC 61850 standard. As such, the communication protocols raise security risks and cyber threats. The power system communication is also to be secured using IEC 62351 as a reference. The industrial environment has recognised various security vulnerabilities to which the power system may be susceptible. The substations are critical power systems that require high input speeds, redundancy and dedicated equipment to meet the operating requirements. The execution of IEC 62351-6 presents the most computational requirements as far as implementation is concerned. It must perform the responsibility of authentication security mechanism of GOOSE and SMV in the required critical transfer time of 3ms. This is paramount and essential for protecting and safeguarding substation functioning.

One of the recommended security mechanisms is encryption. Encryption is used to achieve confidentiality in substation communication. IEC 62351, Part 3, and Part 4 propose the use of TLS for TCP/IP to provide confidentiality. Symmetric key and asymmetric key are the two methods used for encryption. The symmetric algorithm employs one key to encrypt and decrypt, while the asymmetric algorithm utilises two separate keys to encrypt and decrypt.

One of the commonly used symmetric block ciphers is Data Encryption Standard (DES), TripleDES, and Advanced Encryption Standard (AES). One of the most extensively used methods is DES, however, as a result of concerns regarding the level of security (vulnerable to brute force attack) it provided, the TripleDES algorithm was presented to the DES to increase the security level however it presented a slow encryption algorithm (Weerathunga, Pubudu Eroshan, 2012). AES is the new recommended standard.

The most popular asymmetric cipher is the RSA algorithm. RSA is a cryptographic algorithm employed for key exchanges and digital signatures. However, it has the disadvantage of being time-consuming and requires a greater amount of processing power. RSA encryption is also vulnerable to brute force attacks however with the implementation of complex keys the brute force attack can be prevented. Another development would be to use Elliptic Curve Cryptography (ECC) which has a small key size and a computational advantage. Asymmetric encryption ciphers are cryptographically more secure than symmetric ciphers but their encryption and decryption speeds are too slow for power system communication protection (Weerathunga, Pubudu Eroshan, 2012). Therefore, encryption only provides additional CPU load on the IEDs and increases GOOSE transmission time. The communication system must be able to meet the timing demands associated with Protection, Automation and Controls (PAC). A detailed investigation of its behaviour must be taken into account given the time-constrained nature of SAS applications.

1.4.3 Proposed Solution

A secure substation automation control operation is vital due to the critical information system. Standardised communication is critical in power system automation. The standard IEC 61850 is popular (due to its flexible and robust modelling) because it is set to become the standard for power system communication. IEC 61850 specifies protocols that are set to address the problem of compatibility. Of the customs, GOOSE is specifically made to meet the needs of operations, while being mindful of timing requirements. As the number of critical infrastructures employing IEC 61850 increases, cybersecurity aspects are also relevant to the study. IEC 62351-6 and IEC 61850 standards work together and address security mechanisms to protect IEC 61850 messages. These mechanisms must not cause delays in IEC 61850 messages as GOOSE and SV messages have time-critical requirements. Considering the speed, the Keyed Hash-Message Authentication Code-Secure Hash Algorithm (HMAC-SHA256) must be introduced to authenticate GOOSE and SV messages to achieve integrity and authentication of Message Authentication Code (MAC) based digital signature algorithm.

To address the above problem, this study examines the development of a security algorithm and evaluates the impact of the security mechanism on IEC 61850 GOOSE and R-SV communication performance.

Existing cybersecurity features from the open-source code library will be introduced to incorporate the security algorithm (https://github.com/61850security). By using HMAC techniques, it secures GOOSE message transmission. The secure GOOSE functions are created using the C programming language and the OpenSSL library. The proposed security method was shown to be capable of meeting GOOSE's time requirements by (Hussain et al., 2019). The literature review demonstrates that increasing research is being done on protecting GOOSE communication, although applying authentication techniques to SV protocol is still uncommon. This thesis presents a design-based solution algorithm and analysis for implementing a secure R-SV message to prove that the enabled security algorithm can secure SV packets with negligible timing requirements. Most importantly the implementation will conform to the structure of the SV protocol data unit (PDU). The solution will be tested and validated with an HMAC algorithm on a software platform.

1.5 Research Aims and Objectives

1.5.1 Research Aims

This project will explore and evaluate the application of standard IEC 61850 and IEC 62351 technology in the automation environment of the substations.

This study aims:

1. To develop, implement, and analyse various encryption algorithms and key management agreements for GOOSE and R-SV.

1.5.2 Research Objectives

The objectives of this study seek to address:

1. Literature review on the history of substation automation systems.
2. Literature review of data encryption algorithms.
3. Literature review of IEC 61850 and IEC62351 standards.
4. Literature review: To study the IEC 61850 standard methodologies and mapping.
5. Theoretical analysis of data encryption algorithms.
6. Investigation of the effect of messaging propagation delays introduced by the data encryption algorithm.
7. To enhance the compatibility of data encryption algorithms, tailored to meet the requirements of IEC61850 standard-based communication in substations through adaptation and improvisation.
8. Investigation of the performance of the existing and proposed algorithm for various case studies.
9. Development of an algorithm for data transfer on the Ethernet communications network.
10. An algorithm is to be synthesised and ported to a software deployment using a Kali-Linux virtual machine to simulate the algorithm.
11. The development of testing and validation of security algorithm for R-SV message.
12. Analysis of results and conclusion.

1.6 Hypothesis

The implemented authenticated encryption algorithm will comply with IEC 61850 and IEC 62351 and the critical time requirements of data transmission are adhered to. An authenticated encryption algorithm is implemented for maintaining message confidentiality and integrity. The developed security algorithm is utilised on the SV frame structure. The R-SV messages are captured using Wireshark to validate the data traffic and Ettercap is used to perform an MITM attack by spoofing network traffic. Simulation results indicate that the EtM algorithm can be used for GOOSE/R-SV messages while meeting stringent latency criteria and the future IEC 62351 security standards can advocate encryption for R-SV communication.

The research work will test the following:

1. Integration of the HMAC algorithm with an R-SV message.
2. Ensure that R-SV packet frame structure is conformed to, and protocol complies with IEC 61850.
3. Ensure that the HMAC algorithm conforms to IEC 62351-6.
4. Prove that the implemented HMAC algorithm meets timing and security requirements.

1.7 Delineation of the Research

This thesis is to develop an authentication algorithm for IEC 61850 GOOSE and R-SV messages and further evaluate the impact of the security mechanism on IEC 61850 communication performance. The design will be developed and employed on a Kali-Linux software platform; where the client-server code and security algorithm for R-SV will be tested and validated. For GOOSE and R-SV communications, an authentication algorithm will be used to ensure message integrity and authenticity. As such, considerable knowledge is required for raw socket programming and code development for client-server for GOOSE and R-SV message structure.

The following tasks form part of the project:

1. Development of methods for IEC 61850 standard implementation.
2. Development of methods for IEC 62351 standard implementation.
3. Development and implementation of an authenticated encryption algorithm for R-SV messages.
4. A C program running on a personal computer is proposed for the simulation of the R-SV message. The choice of C-language for simulating IEC 61850 Sampled Value streams is not mandated by the standard itself but is often made for practical reasons.

5. C is a low-level programming language. C allows developers to optimise code for performance, ensuring that the simulation runs efficiently. C-language is generally faster and more efficient. It does not increase the payload and the frame structure is compliant.
6. Wireshark is proposed for the capture and analysis of the data packet of the generated R-SV messages.
7. Ettercap is proposed for generating the MITM attack.

The design and implementation will be conducted on a simulation setup for the confirmation of GOOSE and R-SV messages as illustrated in Figure 1.2.

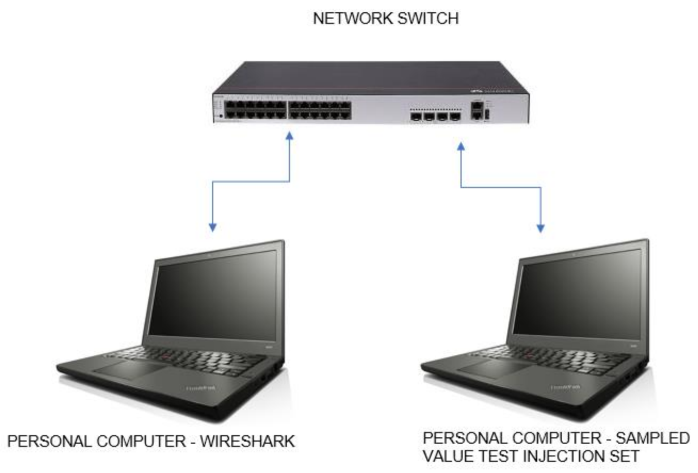

NETWORK SWITCH
PERSONAL COMPUTER - WIRESHARK
PERSONAL COMPUTER - SAMPLED VALUE TEST INJECTION SET

Figure 1.2: Simulation Setup

The thesis studies security mechanism that deploys authentication codes for GOOSE in particular and R-SV communication.

1.8 Significance of Research

Conformance to the IEC 61850 and IEC 62351 is crucial for the optimal operation of the substation automation network. IEC 61850 provides mechanisms for developing the best engineering standard for substation automation systems. Conformance to the IEC 62351 standard is of paramount importance as cyber security is an increasing concern regarding power systems as industrial plants have been compromised by attacks and viruses. IEC 62351 specifies communication protocols and the security mechanisms suitable for protecting information in the substation automation network. The development of cyber security algorithms and techniques for information security to ensure the CIA will contribute to the evolution of substation automation networks. It is important to develop encryption mechanisms such as algorithms and techniques to ensure the authenticity and integrity security of GOOSE and SV communication in a power system.

This study will contribute towards the growing knowledge base in substation automation and cyber security. Furthermore, it will also facilitate the education of technologists and engineers.

1.9 Research Design and Methodology

1.9.1 Research Design

This is an experimental study design that aims to take particular factors into account in accordance with the hypothesis. A brief description of the research plan is presented accordingly:

1. Define Objective: Development of an authentication algorithm for IEC 61850 messages.
2. Planning/Designing Process: The development of the thesis will include the design and implementation phase. The review of the literature and available implemented solutions is to be studied.
3. Experimentation Procedure: The design phase will focus on the code development of the authenticated encryption algorithm adhering to the frame structure for GOOSE/R-SV messages. The implementation phase is the deployment of the presented authenticated encryption algorithm using the Kali Linux command line to run the code.
4. Analysis/Modelling: Wireshark is used to collect and analyse data, and Ettercap is used to launch MITM attacks to infect traffic between publisher-subscriber devices.
5. Interpretation of Results: The Wireshark capture will show all the required fields according to IEC 61850-8 for GOOSE, IEC 61850-9-2 for R-SV and the generated attacks of the packets are analysed. This will provide a platform that can be easily reconfigured to test improvisations or updates and amendments to programs.
6. Conclusion: Questions must be resolved. Can authenticated encryption be successfully implemented with R-SV PDU? Does the implemented security algorithm meet timing requirements? Was the frame structure for GOOSE/R-SV compliant?

1.9.2 Methodology

A review of the literature will be undertaken to ascertain the trend in IEC 61850 and IEC 62351 for GOOSE and SV messages. Attention will be focused on works related to theoretical studies and practical implementations of proposed security mechanisms for GOOSE and SV messages. Newly proposed encryption methods will also be a focal point to ascertain mechanisms for best practice. The article uses an experimental research design to develop and evaluate authentication algorithms for IEC 61850 GOOSE and R-SV messages in substation automation systems. The article follows these steps:

1. Review the literature on substation automation systems, communication and cyber security standards, authentication methods and data encryption algorithms.
2. Analyse the theoretical aspects of authentication methods and data encryption algorithms and their impact on message propagation delays.
3. Develop an algorithm for data transfer on the Ethernet communications network using C programming language and OpenSSL library.
4. Simulate the algorithm on a personal computer using Kali-Linux virtual machine, Wireshark, and Ettercap tools.
5. Test and validate the security algorithm for R-SV messages and evaluate its performance for various case studies.
6. Conclude with a discussion of the results and future research directions.

1.10 Organisation of the Thesis

The thesis is divided into six chapters detailing the introduction and background into substation automation systems, problem definition, a literature review into substation automation trends, a detailed study into IEC 61850 and IEC 62351standard, an investigation into the IEC 61850 GOOSE and Sampled value messaging system, and the results of the software development of the simulated security algorithm for GOOSE message.

Chapter 1: presents an overview of the substation automation developments, IEC 61850, and IEC 62351 standards and highlights the problem definition, project aims and objectives, and research methodologies.

Chapter 2: presents the literature search and literature review into IEC 61850 and IEC 62351 standards. This section also reviews the GOOSE and SV messages. The development of security mechanisms in substation automation. Various technical papers, journals, and articles were read and analysed.

Chapter 3: discusses comprehensively an overview of the IEC 61850 standard with emphasis on sampled value messaging and in particular focus on GOOSE messaging.

Chapter 4: presents various authentication codes for GOOSE and discusses the output of each code. The codes will be compiled using Kali-Linux Virtual Box.

Chapter 5: describes the authentication algorithm development and implementation of the GOOSE message and provides snippets of the code. The design will conform to the applicable standards, IEC 61850, and IEC 62351.

Chapter 6: provides a conclusion to this research project and provides expansion prospects such as real-time implementation for future research projects. Following Chapter 6, the references and appendices are supplied.

1.11 Journal Papers

A journal paper will be published.

1.12 Artifact

The project will contribute towards an integrated merging unit that can be deployed in a substation environment.

1.13 Conclusion

This chapter discussed the necessity of developing and implementing cyber security in the substation automation environment. Traditional communication protocols need to be updated with security measures that can counteract current security threats. The electricity grid is no longer contained within its physical structure, which makes security even more difficult. If security standards are not updated, energy data could become even more exposed since it is transmitted through and stored in the cloud.

A brief overview of the communication and cyber security standard was discussed. Furthermore, the introduction to Hash Message Authentication Code (HMAC) was deliberated.

This chapter outlines the research project's goals and objectives, problem statements, methods, and contributions and further includes a chapter outline for this thesis.

The second chapter contains an extensive literature review of research projects on the development of algorithms for secure GOOSE and R-SV messages. This chapter of the literature review also covers the introduction of the IEC 61850 standard, the IEC 62351 standard, and raw socket programming. Several papers are reviewed to assess/outline the research aims and objectives.

2. CHAPTER TWO: LITERATURE REVIEW

2.1 Introduction

As the grid gets smarter and more networked, substation automation relies on digital control and digital communication to secure, control, and monitor the grid's operations. The evolution of Substation Automation Systems (SAS) has continuously sought advanced technology integration to deliver power with the protection of operators, power system equipment, and the end-user. The stability of the grid is vital as electricity is to be delivered continuously without failures. This motivated the need for quality and reliable electricity supply. The intermittent interruption of communication in the grid, the growing concerns of standardisation in the substation automation environment, and cyber security have gained attention to address stable and reliable communication and cyber security in the power system. Hence, the necessity for standardisation of communication networks, and substation design, prompted the rapid development and implementation of technology to networked smart grid systems. Communication is the fundamental factor in SAS, and virtually the security of the power system environment must be developed with performance and reliability addressed.

Concerns regarding the security of SAS communication are studied. Also, the development of various security mechanisms for a secure power system is addressed. The substation is a critical infrastructure, consisting of critical devices that should be protected against malicious security attacks. Unsecured, and unprotected communication networks raise various security issues. IEC 61850 and IEC 62351 are receiving worldwide consideration and as such have managed to become the substation automation communication and cyber security standard of the future. The IEC 61850 GOOSE and SV messages are essential for the secure functioning of the electrical network.

The following is a list of the sections that make up this chapter: Section 2.2 discusses the awareness of the problem in the literature. A review of relevant literature about cybersecurity is found in section 2.3. Section 2.4 focuses on IEC 61850-8-1 GOOSE and IEC 61850-9-2 SV messages. Furthermore, it presents a literature review relevant to the mapping, publishing, evaluation, and application of IEC 61850 communication in a SAS. Section 2.5 deals with the literature review of the cyber security standard, IEC 62351 and section 2.6 provides an overview of socket programming. Section 2.7 draws the conclusion.

2.2 Awareness of the Problem in Literature

When the internet and the advancement of digitalisation are used to facilitate communication between literally billions of people and are used as a tool for commerce, social interaction, and the exchange of an increasing amount of personal information, security has become an extremely important issue for all internet users to deal with daily. Smart grids can be connected to substations via the internet (for remote monitoring and control, fault detection, cybersecurity monitoring, grid optimisation, and advanced metering infrastructure), as well as to other smart grids. As a result, energy generation may now be dispersed throughout the entire grid, which is equipped with sensors to assist in monitoring, protecting, and controlling it. Smart grids and microgrids, in addition to their advantages, have some disadvantages and additional challenges to overcome, such as concerns about compromised data security and privacy. As a result of communicating through a network, security must be strengthened to avoid being targeted by hostile activities and to maintain a stable grid. Data protection professionals are concerned about the amount of information that can be collected and provided to service providers, as well as the chance that this information could end up in the hands of malicious attackers. These data sets may contain significant information on the subject (Sontowski, 2016). Security and privacy are two crucial needs for smart grids, yet they are also important for traditional grids. Because a grid contains a large number of security-critical systems, it is vital to ensure their safety and security. When it comes to smart grids, the protocol IEC 61850 is widely employed. However, IEC 61850 does not have any standardised security procedures. Techniques for ensuring secure communication can be implemented. Therefore, the communication standard can be used in conjunction with TLS and SSL. These two protocols are responsible for network communication security (Sontowski, 2016). Part 3 of the IEC 62351 standard was introduced to address security measures. It was proposed in IEC 62351-3 and IEC 62351-4 that TLS be used for profiles that contain TCP/IP. TLS and IPSec encrypt data to maintain the security and privacy of communications (Weerathunga, Pubudu Eroshan, 2012).

2.3 Theoretical Framework – Data Security and Privacy

Inadequate data security costs organisations billions of dollars per year, according to industry estimates (Bogdanov et al., 2011a). As such, a significant amount of money is spent to safeguard the systems after they have been compromised. Significantly so, it is now possible to decrease numerous hazards and improve the security of systems by implementing security procedures. Security measures must be put in place from the beginning of a project, and they must be in place throughout the entire process, from design to development to implementation and they must be tested regularly. Taking enhanced security measures and precautions is critically important. Many issues can be traced back to a lack of proper data security and data privacy measures in place. Most businesses place a high value on privacy to maintain their consumers' trust. Privacy, on the other hand, is a considerably broader term.

(Hitachi ABB, n.d.) design the FOX615 multiplexing platform's TEGO1 interface board. TEGO1 has an integrated IEC 61850 interface that allows it to connect directly to an electrical substation's IEC 61850 station/process bus. This interface enables the use of GOOSE and SV messages for line distance and differential protection, as well as new applications like remote trip and interlocking. TEGO1 is included in IEC 61850. It portrays the distant IED in the local substation. TEGO1 was created to address one of three major issues: cybersecurity. Because GOOSE/R-SV messages are based on Ethernet packets and the station bus is based on an Ethernet Switched network, cyber security aspects such as data integrity and access limitations must be considered, particularly for important applications needing high real-time performance. Furthermore, TEGO1 enables GOOSE/SV message authentication for man-in-the-middle (MITM) attacks and replay protection. Furthermore, crucial features such as redundancy in the event of a communication breakdown, filtering capabilities, and translation capabilities are provided. To maintain privacy and security in computing, using encryption to protect one's data is extremely crucial. Encryption is one of those cyber security mechanisms that are frequently in the news, especially when it comes to government agencies. Encryption is at the heart of Internet security and privacy protection (Thakkar, 2020). Considering the increasing interconnection of computer networks and the sophistication of cyber-attacks, cryptography is becoming an increasingly important tool for ensuring that data users can maintain their privacy and confidentiality, while also ensuring authentication, integrity, availability, and identification (Hamouda, 2020). Cryptography methods are critical in protecting data from hostile attacks. It is possible to divide and distinguish the encryption algorithm into two types: symmetric key (private key) and asymmetric key (public key). The public key is used to encrypt the communication, while the private key is used to decrypt the message once the message has been encrypted. The encryption process is made feasible using cryptographic keys in conjunction with encryption methods, which are discussed further in Table 2.1 (Thakkar, 2020). The algorithms must be designed in such a way that they are difficult to be cracked by intruders. This list of commonly used encryption algorithms covers algorithms such as Rivest–Shamir–Adleman (RSA), Elliptic Curve Cryptography (ECC), Triple Data Encryption Standard (3DES), Advanced Encryption System (AES), and others (Rivest Cipher). The DES and AES algorithms are the most well-known of these encryption algorithms. The analysis and construction of these protocols must be done efficiently to ensure the secrecy of the messages that are being conveyed.

An overview of the encryption techniques is presented in Figures 2.1, 2.2, and 2.3 respectively. Essentially Figure 2.1 presents the symmetric encryption. The key point in symmetric encryption is that both the sender and receiver share identical keys. The sender wants to communicate with the recipient by sending a message (plaintext). The sender will take the secret key that will be used to encrypt the message (ciphertext) and will transfer it across the internet. The message (ciphertext) will be received by the recipient using the same secret key, who will decrypt it (plaintext) and be able to read it.

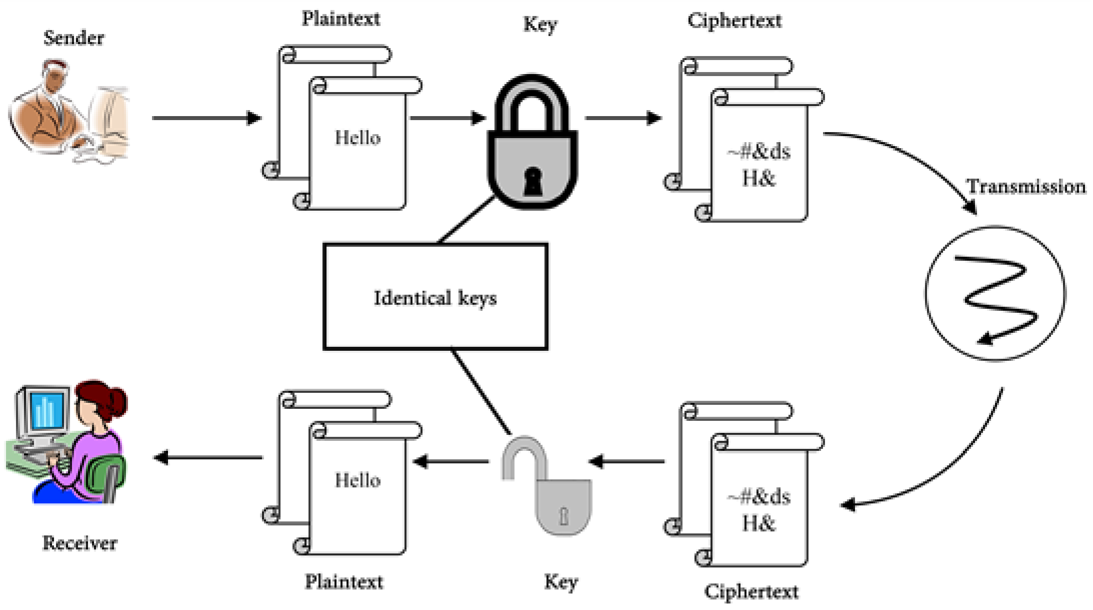


Figure 2.1: Symmetric Encryption Example (Hamouda, 2020).

Asymmetric encryption is presented in Figure 2.2. The public key and the secret key are used for encryption. The public key is used for encryption and is freely distributable. The secret key is used to decode the data and is only known to the key pair's owner. In this situation, the client encrypts the message (plaintext) using the public key. This is critical because it assures that only the intended recipient with the matching private keys can view the messages, not an unauthorised user. The server then uses the secret key to decrypt the message (ciphertext).

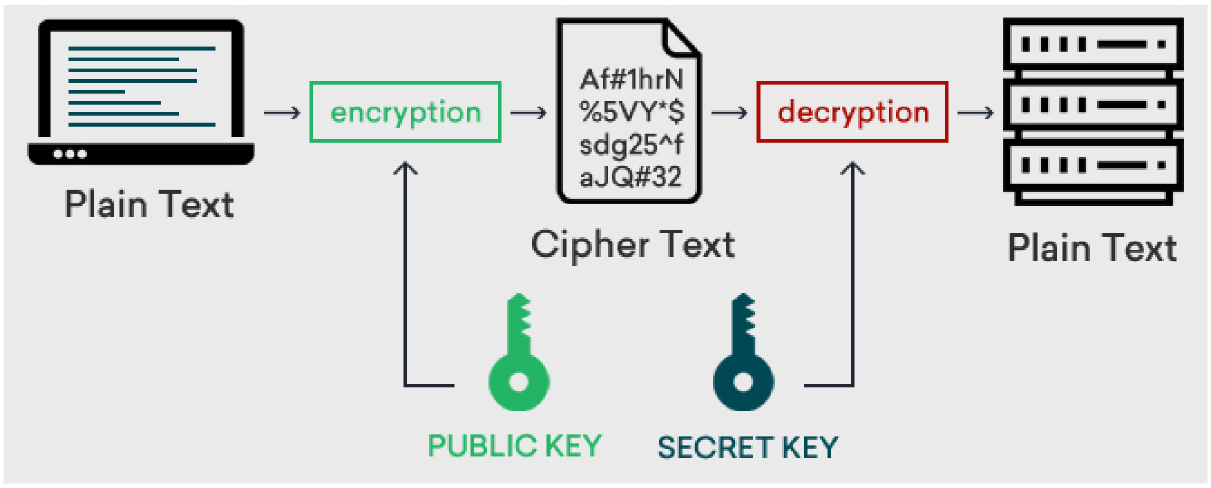


Figure 2.2: Asymmetric Encryption Example (ClickSSL, 2022)

Furthermore, hybrid encryption is a combination of symmetric and asymmetric cryptography. The goal is for the sender to encrypt the symmetric method's secret key using the asymmetric method's public key and then deliver this data to the recipient. Both the encrypted symmetric key and the encrypted data are sent to the receiver. The receiver decrypts the secret symmetric key, and both parties now have the key for symmetric encryption, allowing the sender to quickly transfer the remainder of the data.

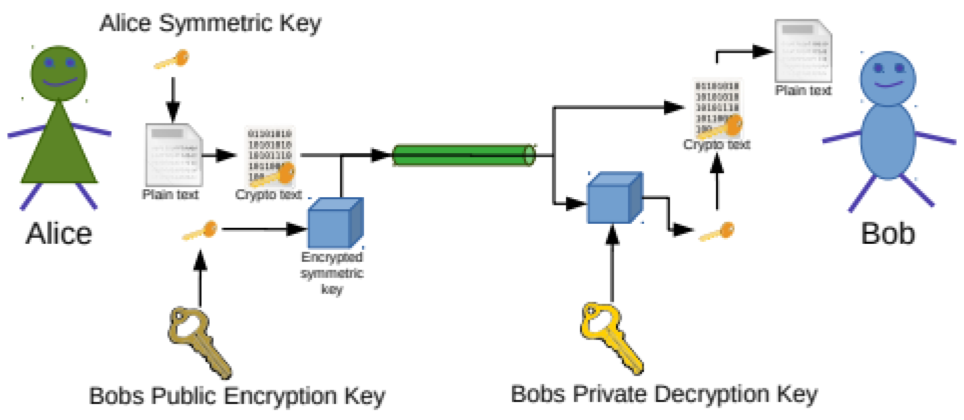


Figure 2.3: Hybrid Encryption Example (Sontowski, 2016)

When it comes to data security, encryption is a key component of cryptography, which is the most effective and widely used approach available today (Pedamkar, 2023). Both symmetric and asymmetric encryption have their own set of advantages, and we cannot choose one over the other because they are mutually exclusive. Asymmetric encryption, on the other hand, is unquestionably superior in terms of security because it assures authentication and non-repudiation. However, performance is a factor that we cannot afford to overlook, which is why symmetric encryption will continue to be required.

Aspects of security and applications abound, ranging from safe commerce and payments to private communications and the protection of healthcare data, among other things. When it comes to secure communications, encryption is a critical component of the equation. Although cryptography is necessary for data security, it is not adequate on its own. Encryption can provide confidentiality, but it is incapable of providing integrity. To achieve integrity, authentication must be used in conjunction with encryption technology. Using a covert channel, the client can transfer data to the server without being detected (S.A. Fatayer, 2020). Before using the covert channel, both parties must have already shared information. The confidentiality of pre-agreement information and the detectability of covert channels are the two most significant issues for covert channels (S.A. Fatayer, 2020). (S.A. Fatayer, 2020) provides a demonstration of how integrating encryption, authentication, and a covert channel results in a new covert channel that provides security of data being transmitted while remaining undetected. The covert channel technique is shown in Figure 2.4. The proposed covert channel requires shared information between the client and server. The technique needs a pre-shared table that consists of the original keys and their corresponding fake keys. Each original key has multiple fake keys. In this case, the most important feature is that the bogus key is utilised in the communication channel, but the original key is kept confidential on both sides (client and server). The secrecy of the information between the client and server was ensured by the use of an encryption algorithm and the HMAC algorithm is used to verify integrity (S.A. Fatayer, 2020). Covert channels together with encryption and authentication lead to secure communication.

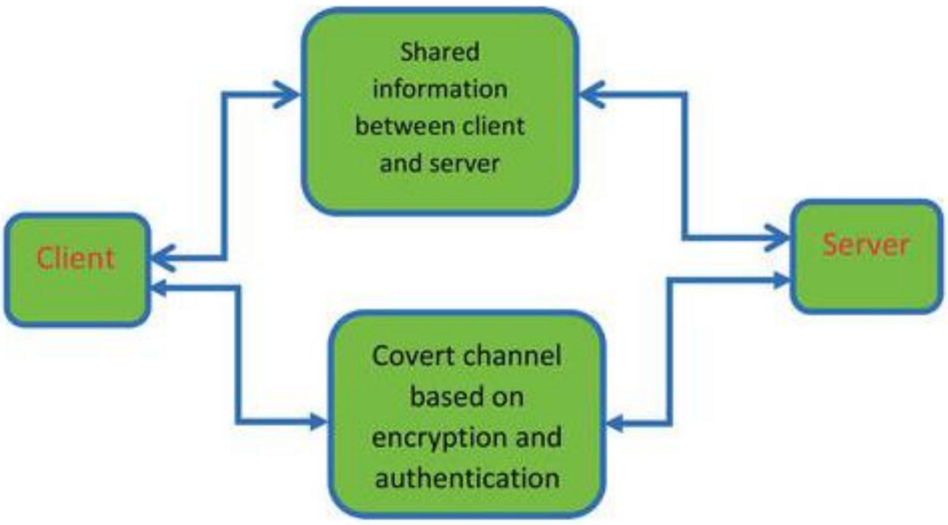

Shared information between client and server
Client
Server
Covert channel based on encryption and authentication

Figure 2.4: Covert channel secure communication (S.A. Fatayer, 2020)

The author further presents the characteristics and properties of the covert channel stating that for security and more private communication, plausibility, undetectability, and indispensability need to be achieved. This technique addresses secrecy requirements through encryption, as well as integrity requirements using an authentication algorithm. A secure communication channel between the client and the server that allows them to communicate data safely and to agree on keys that will be used for future communication is established between them. Table 2.1 discusses the encryption methods.

Table 2.1: Encryption Techniques (Thakkar, 2020)

Type of Encryption	Method	Types	Advantages	Disadvantages	Usage
Symmetric Encryption	Data is encrypted and decrypted with the help of a single cryptographic key. Ideal for applications in which a large amount of data needs to be encrypted regularly. Symmetric encryption is primarily used for encryption.	Data Encryption Standard (DES)	- The simplicity with which symmetric encryption is implemented is its most distinguishing characteristic. As such, symmetric encryption is faster, requires less computational power, and does not dampen internet speed.	- Low encryption key length. - It was cracked by many security researchers; it is officially no longer in use and replaced by the AES algorithm.	- SSL/TLS protocol (website security)
		Triple Data Encryption Standard (3DES) - 3DES was created to address some of the shortcomings of the DES algorithm.	- 3DES is significantly more difficult to crack than its predecessor, DES. Each data block is subjected to three repetitions of the DES algorithm.	- According to Karthikeyan Bhargavan and Gatan Leurent of Inria (Paris), attacks against TDEA were analysed and implemented in real-world applications. The results show that the	- SSL/TLS protocol (website security)

				collision attack on TDEA represents a serious security vulnerability for many common uses of these protocols, including the HTTPS protocol for secure Internet connections. Furthermore, the investigation reveals that security vulnerability continues to be a severe concern. After 2023, the use of 3DES will be phased out in all new software applications.	
		Advanced Encryption System (AES) - It is one of the most widely used forms of encryption algorithms, and it was developed as a replacement for the DES method.	- AES is a secure, quick, and versatile encryption method. - AES is a significantly faster algorithm. - The most significant benefit is the longer the keys are, the more difficult it is to crack them.	- Finding a weakness in the algorithm. - Brute force search. - XSL (extended sparse linearisation) attack.	- Wireless security -Processor security and file encryption -SSL/TLS protocol (website security) -Wi-Fi security -Mobile app encryption -Virtual private network (VPN) etc.

Asymmetric Encryption	Entails the use of several keys for the encryption and decryption of information. An asymmetric encryption scheme consists of two different encryption keys that are mathematically related to one another. Asymmetric encryption provides the benefits of encryption, authentication, and non-repudiation all in one package.	Ron Rivest, Adi Shamir, and Leonard Adleman (RSA) Encryption	- Data remains protected against man-in-the-middle (MITM) attacks. - Provides authentication. - Provides scalability. Various encryption key lengths can be used. - Most widely used asymmetric encryption algorithm	- RSA is a deterministic method, it is possible to conduct a passive attack on it as well. To prevent these attacks, a hash function can be incorporated into the encryption function that allows the receiver to confirm whether he got a valid message.	- website security - email encryption - crypto-currency
		Elliptic Curve Cryptography (ECC) Encryption	- Data remains protected against man-in-the-middle (MITM) attacks. - Provides authentication. - It's impossible to crack as it is a complex algorithm. - Faster performance since less networking load and computing power.	- Many server software has not added support for ECC SSL/TLS certificates. RSA continues to be widely used.	- website security
Hybrid Encryption - Symmetric + Asymmetric Encryption	It combines the best features of both symmetric and asymmetric encryption methods, resulting in a synergistic effect that allows for		- Fast communication - Less computing power		- website security - web browsers - E-mail encryption

	the development of resilient encryption systems. As long as both the public and private keys are kept completely safe, this sort of encryption is regarded to be extremely secure.				

As encryption, HMAC is a fundamental security procedure. Because of the rapid evolution of communication standards, security has become an increasingly important requirement, particularly in today's world. It is vital to maintain secrecy however, without authentication, we are unable to identify and authenticate the persons or parties involved in a transaction. Encryption and HMAC are critical in addressing security issues. The implementation of these security techniques needs to be employed in critical infrastructures and provisions developed in future technologies. HMAC has been established as a standard for robust authentication with additional security features.

The Message Authentication Code (MAC) is used to ensure the authenticity and integrity of messages. Most MACs use symmetric key techniques and one-way hash functions. AES-CBC-MAC employs a symmetric mechanism, whereas HMAC utilises a hash function-based technique. The first release of the secure hash function algorithm was known as SHA-1. The second generation of the SHA algorithm, SHA-2, has been created to take advantage of the enhanced security features of the Advanced Encryption Standard (AES) protocol (AES). Brute force attacks on the MAC are more challenging to execute than detecting a collision in a hash function, and the cryptographic strength of AES block cipher encryption is higher as well. The AES-Cipher Block Chain (CBC)-MAC algorithm is therefore regarded as a safe authentication method (Weerathunga, Pubudu Eroshan, 2012). The hash algorithm SHA-1 provides the lowest level of security. SHA-2 hash algorithms are being given more attention as a result of the increase in the number of security bits associated with digital signature algorithms and AES and as a result, they are better suited for dealing with recent developments in computer security.

To evaluate HMAC over the message or file, the following expression is required to compute (Dondossola & Terruggia, 2015).

$$MAC(text)t = HMAC(K, text)t = H((K_0 \oplus opad)|| H((K_0 \oplus ipad) || text))_t$$

Equation 2.1

HMAC uses the following parameters and symbols (Dondossola & Terruggia, 2015):

“B	=	*Block size (in bytes) e.g., 64 bytes = 512 bits,*
H	=	*Approved hash function (SHA-1, SHA-2),*
ipad	=	*Inner pad e.g., the byte x36 times repeated B times,*
opad	=	*Outer pad e.g., the byte x’5c’ repeated B times,*
K	=	*Secret key shared between the sender and the receiver,*
K_o	=	*The key K with zeros appended to form a B byte key,*
L	=	*Block size (in bytes) of the output of the approved hash function,*
T	=	*The number of bytes of MAC,*
text	=	*The data on which the HMAC is calculated; the length of the data is n bits, where the maximum value for n depends on the hash algorithm used.”*

The HMAC method is presented in Figure 2.5. Table 2.2 depicts the algorithm's process operation (Dondossola & Terruggia, 2015).

Table 2.2: HMAC Algorithm

STEPS	DESCRIPTION
1.	*“If the length of K = B, set K_0 = K. Go to step 4”*
2.	*“If the length of K > B, hash K to obtain an L byte string: K = H(K).*
3.	*“If the length of K < B, append zeros to the end of K to create a B-byte string K_0 (e.g., if K is 20 bytes in length and B = 64, then K will be appended with 44 zero bytes 0x00).”*
4.	*“Exclusive-Or K_0 with ipad to produce a B-byte string:* ***$K_0 \oplus$ ipad****.”*
5.	*“Append the stream of data 'text' to the string resulting from step 4:* ***($K_0 \oplus$ ipad) || text”***
6.	*“Apply H to the stream generated in step 5:* ***H(($K_0 \oplus$ ipad) || text).”***
7.	*“Exclusive-Or K_0 with opad:* ***$K_0 \oplus$ opad****.”*

8.	*"Append the result from step 6 to step 7:* ***$(K_0 \oplus opad)$ || $H((K_0 \oplus ipad)$ || $text)$****."*
9.	*"Apply H to the result from step 8:* ***$H((K_0 \oplus opad)$ || $H((K_0 \oplus ipad)$ || $text))$****."*
10.	*"Select the leftmost t bytes of the result of step 9 as the MAC."*

The hash method and key are used on both the sender and recipient sides to obtain the matching HMAC value, which is used to verify the authenticity of the data. As mentioned above, HMAC makes use of a cryptographic hash function that is irreversible; hence, when we utilise HMAC from the sender side to encrypt a message using the HMAC formula, the message is encrypted at the sender side (Gupta et al., 2017). The hash function and the key will be used by the receiver to produce a value that is equal to or greater than the hash. We decrypt the cyphertext with the help of an authentication key and compute the HMAC on the plain text. If both values are equivalent, the decryption is accepted; otherwise, the decryption is rejected.

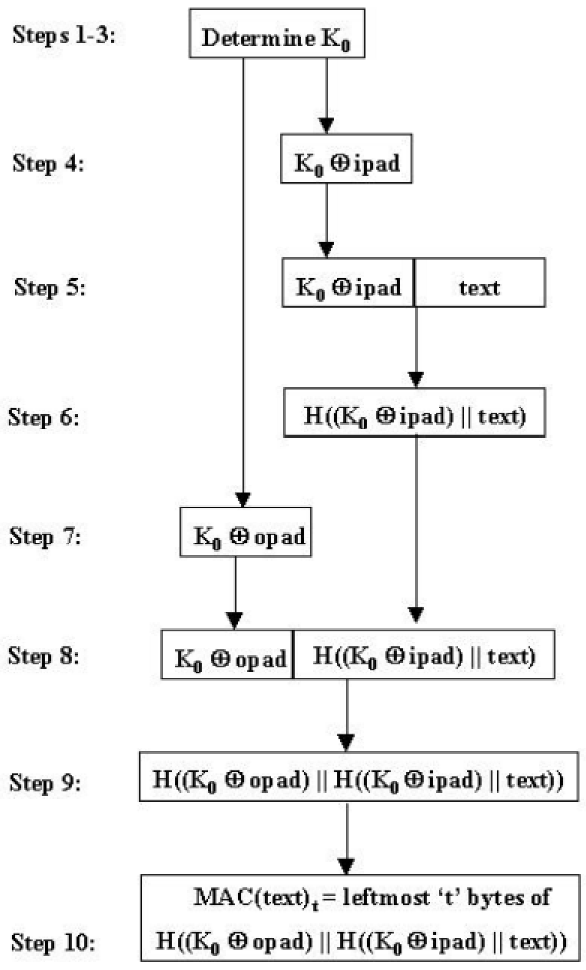


Figure 2.5: Illustration of the HMAC Construction (Gupta et al., 2017)

Known as a cryptographic algorithm, the HMAC provides the maximum level of protection against security threats. HMACs are used to exchange information between two parties who are both aware of the secret key used to encrypt the information. A digital signature does not require the verification of a secret key to be authenticated. In cryptography platforms and other industries, encryption and hash methods have important applications. Encryption is often used to protect the confidentiality of data. Only authorised people with the key should be able to access the data. On the other hand, hashing works well for verification; knowing the actual data is unnecessary, just whether or not the hashes are the same. Table 2.3 shows reviewed papers for data security.

Table 2.3: Data security reviewed literature papers.

Paper	Research Objectives	Method	Outcomes
(Bhanot & Hans, 2015)	The authors conduct a comparative analysis and review of several encryption techniques.	Different Algorithms are compared based on various parameters. The authors looked at ten different data encryption techniques, including DES, Triple DES, RSA, AES, ECC, BLOWFISH, TWOFISH, THREEFISH, RC5, and IDEA, amongst other things.	Each algorithm has its own set of advantages that vary depending on the parameters used. The strength of each encryption technique is determined by the key management system used, the type of cryptography employed, and the number of keys used. Even though there is significant room for improvement, the authors propose the ECC and Blowfish encryption algorithms. The authors reiterate that these encryption algorithms are leading when it comes to security level and providing faster encryption speed.
(Alajbegović et al., 2006)	This paper presents the DSA security technique.	Mathematica 4.0 will be used to show the production and verification of signatures using DSA, as	We may utilise the DSA to generate digital signatures for any type of message that we send out (encrypted or

		demonstrated by the authors. It is necessary to produce a digital signature as well as a public key (which includes the secret key). DSA is reliant on the confidentiality of the private key is maintained.	not). As a result of its dependence on the discrete logarithm problem, which is extremely difficult to solve, this approach has proven to be extremely safe.
(Michail et al., 2004)	In this research, the authors describe an efficient implementation of the HMAC using the SHA-1 hash function that is both fast and secure in terms of performance. This technique, in conjunction with a shared secret key, is used to authenticate messages sent across the network.	The proposed system architecture has been described in VHDL. The entire system has been thoroughly tested and confirmed using commercial simulation tools, and its functioning has been thoroughly tested and verified. All the internal components in the design were created using XILINX FPGA chips, which are available for purchase online.	The results of the simulations, which were carried out with commercial tools, confirmed the efficiency of the HMAC implementation in terms of both performance and throughput. Special effort has been taken to ensure that the suggested implementation does not add additional design complexity, while at the same time ensuring that functionality is maintained at the needed levels.
(Chen & Yuan, 2012)	In this research, we introduced a Key Derivation Function (KDF) technique based on the HMAC-SHA-256 cryptographic algorithm in LTE networks.	The SHA-256 and HMAC algorithm overview and implementation process are presented.	HMAC-SHA is a hash function utilised in communication and shared key authentication. It is capable of effectively preventing data from being intercepted and tampered with during transmissions; maintaining data integrity, dependability, and security; and preventing data from being compromised.

(Oyelade et al., 2015)	An encryption or decryption system for text data has been developed by the authors, and it makes use of both the DES and the RSA cryptosystems.	It was decided to utilise the asymmetric method for the key encryption and decryption process since it allows for the delivery of keys over a secure channel, whilst the data to be communicated will be encrypted and decrypted using the symmetric technique.	This system was created to achieve a variety of security aspects, including authentication, confidentiality, integrity, and non-repudiation of information. In addition, we combined the speed and strength of the symmetric method with the robustness and key management capabilities of the asymmetric algorithm, resulting in a better encryption algorithm, and we used text data as our experimental data to demonstrate this.
(Hamouda, 2020)	In this article, a comparative analysis of various encryption methods is provided.	The author has conducted a performance analysis of the following encryption techniques: DES, 3DES, and AES, and compared their performance. They have been evaluated based on their capacity to secure data, the time it takes to encrypt data, and the amount of throughput required by the algorithm.	The findings of the comparative analysis demonstrated the capabilities of each algorithm. It concluded that the AES algorithm outperformed all other commonly used encryption algorithms in terms of performance. It was decided to consider security.
(Francis & Monoth, 2018)	A thorough examination of the numerous hybrid cryptosystems presented by various researchers is carried out, with the results being a list of the characteristics of the	Ultimately, the purpose of this research is to investigate and evaluate the advantages of hybrid cryptography. It is critical to ensure the security of information exchange through the internet as well as the	Using the notion of hybrid cryptography, this work adds to the knowledge of cryptography by analysing several algorithms that make use of the concept. It may be argued that,

	algorithms employed in hybrid cryptography.	local storage of secret information. Some of the most important algorithms used in hybrid cryptography, such as the DES, AES, RSA, ECC, and DSA, are examined in detail.	despite the increased computational difficulty, cryptographic goals such as secrecy, integrity, and authenticity can be attained using hybrid cryptographic systems.

Sections 2.4 and 2.5 will review the literature on the use of IEC 61850 and IEC 62351 in automation systems, as well as study the performance of devices that support GOOSE and SV.

2.4 Literature Review – IEC 61850

2.4.1 Introduction

Over the years, communication networks have become a crucial part of the substation automation system (SAS) to provide control and protection in the power system. The advancement of technology has transformed digital communication to be introduced in Data Acquisition Systems (DASs) for monitoring and controlling processes (Ncube, 2012). However, legacy protocols had the disadvantage of limited, and low bandwidth (Sun et al., 2012). As such, motivation was brought forward to improve the reliability of the DASs by improving the bandwidth. This motivated the development and implementation of hardware and software such as microprocessors to enhance control and monitoring procedures. This is further attributed to the integration of IEDs (Ncube, 2012; Sun et al., 2012). GOOSE communications are usually small in size and sent as multicast messages. They are intended for speedy and efficient communication, with relatively minimal bandwidth requirements when compared to SV messages. SV messages contain sampled values, and the amount of data generated is determined by the sampling rate and number of measurement points. As a result, SV may have higher bandwidth requirements than GOOSE. Because GOOSE communications are small and targeted, they have a low potential to congest the bandwidth. Congestion can arise when there are a significant number of events or devices communicating at the same time. SV messages have the potential to cause bandwidth congestion, particularly in situations when a rapid rate of sampling a large number of measurement points is occurring. For the transmission of SV data, latency and bandwidth are crucial parameters. (Groat et al., 2023) examine the communication bandwidth used by GOOSE and SV messages and further provide recommendations on reducing bandwidth utilisation and using the available bandwidth efficiently. The authors present communication capacity utilisation calculations for common applications that are scalable for the number of devices, based on Ethernet frame structure and settable transmission rates. The authors confirm that the sampled value bandwidth exceeds the GOOSE bandwidth.

An IED is a software-configurable microprocessor-based relay with a combination of functions to monitor, protect and control. As a result, combining numerous operations into a single device reduces cost, and time, and increases substation automation levels (Emmanuel, 2014). Before the introduction of IEC 61850 and microprocessor-based relays, legacy communication protocols introduced issues where there was no interoperability between devices and no interchangeability. However, at the time, there was no standard to address such issues. Therefore, a standard needed to be introduced to implement standardisation and provide interoperability across various IEDs.

As such, the IEC 61850 standard was released in its initial version to provide interoperability, a consistent design process (including system setup), and interchangeability. The IEC 61850 standard was first issued in 2002 by Working Group 10 (WG 10) of IEC TC 57. IEC TC 57 is in charge of the development of international standards in substation automation. The specification was created to address issues with older traditional legacy communication protocols. The standard establishes a communication interface between process equipment, bay level, and station level levels of the power system. Figure 2.6 shows a standardised SAS. The standardisation of IEC 61850 came as a need as legacy communication interfaces lacked interoperability, and the deployment and development costs were expensive and high (Emmanuel, 2014).

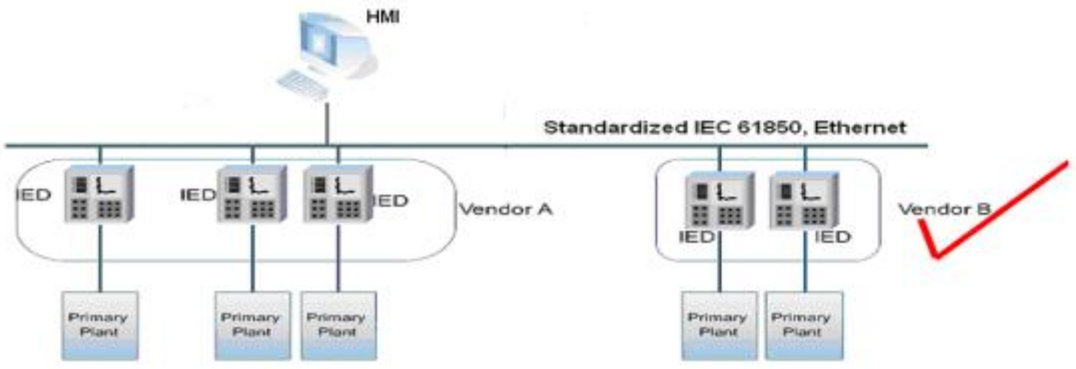


Figure 2.6: Standardised substation automation system (Emmanuel, 2014).

The IEC 61850 standard takes into account communication requirements within the substation automation network. The standard defines the interoperability between functions and devices; the capability of IEDs from various manufacturers to interface with each other for information exchange and use the information for functions providing protection, monitoring, control, and automation. Furthermore, the Ethernet-based communication standard brought benefits of reduced configuration, installation, and commissioning costs and increased power system stability (Karnati, 2020). Based on protocols and standards, IEC 61850 is the ideal communication standard for substation automation purposes (Hohlbaum et al., 2010). This standard aims to establish a level of communication, which supports emerging technologies and meets operational and performance requirements.

Two groups of communication interfaces are available in the IEC 61850 standard: the client-server and peer-to-peer architecture. A client-server architecture enables services like Reporting and Remote Switching. The peer-to-peer architecture supports Generic Substation Event (GSE) services. GSE is connected with time-critical, rapid and reliable communication amongst IEDs. GOOSE messages are associated with the GSE service. The use of the GOOSE message is quite critical in a substation, for the protection of the power system. As such, the GOOSE message structure is important for detecting and isolating any faults in the system.

Part 6 of IEC 61850 describes the use of substation automation SCL in configuration tools for users to configure IEDs. Part 7 of the standard describes the object modelling approach, logical node and data classification, specific object definitions and descriptions, and abstract communications service interface (ASCI). Part 8 describes communications across the station bus, the Local Area Network (LAN) connecting the IEDs and the relay room. Part 9 describes, among other things, communications across the process bus, the LAN connection to the high voltage yard for voltage and current sampled readings, power equipment status report, etc. (Julie, 2014).

One of the key features of IEC 61850 is the separation of the application from the communication through an abstract communications service interface. As illustrated in Figure 2.7, the stack selected according to the highest development technology comprises of Manufacturing Message Specification (MMS) connection layer over Transport Control Protocol/Internet Protocol, TCP/IP. The GOOSE and SV operations run with high-speed switched Ethernet data frames excluding processing of any middle layers.

Fast messages include GOOSE commands such as trip, interlocking, and inter-trip signals. IEC61850-5 offers multiple performance levels for raw data packets from digital equipment for SV communications.

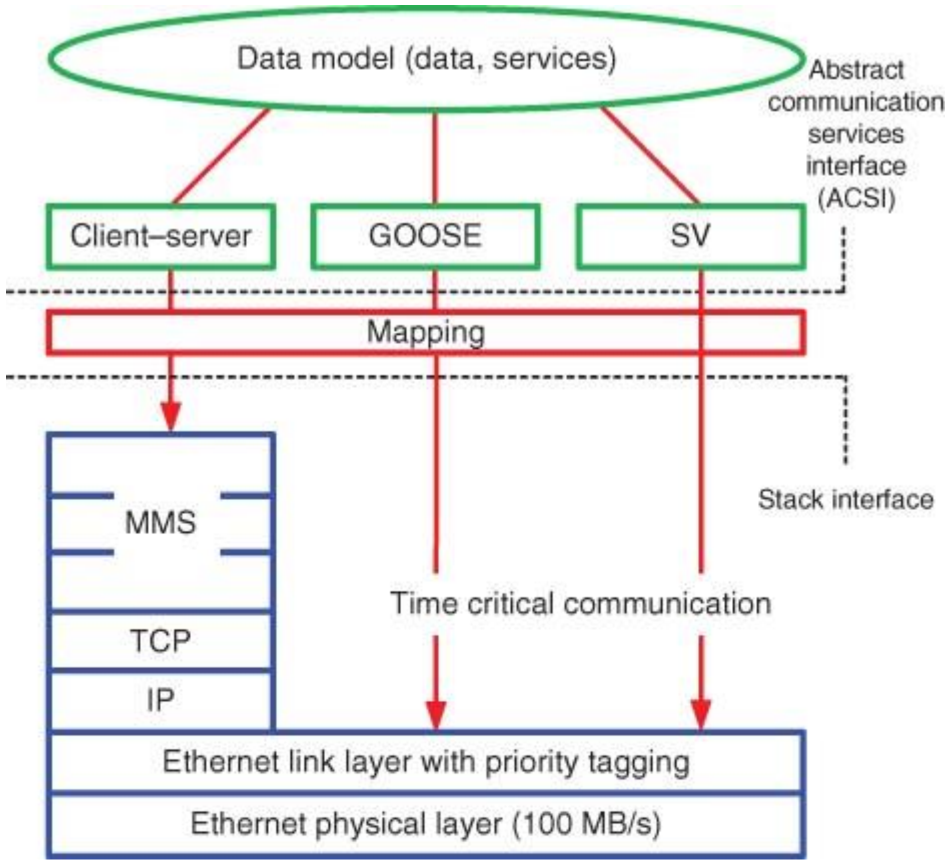

Data model (data, services)
Abstract communication services interface (ACSI)
Client–server
GOOSE
SV
Mapping
Stack interface
MMS
TCP
IP
Time critical communication
Ethernet link layer with priority tagging
Ethernet physical layer (100 MB/s)

Figure 2.7: IEC 61850 Communication Service Overview (Committee et al., 2017)

In substation automation, three levels occur the process, bay, and station level. The process level contains electrical equipment (power transformers, circuit breakers, switch disconnectors, etc). These high-voltage devices connected to the bay level are installed below the process level. Transferred information includes analogue input and output data incorporating current and voltage transformer outputs, as well as trip signals from protection relays. Figure 2.8 depicts the logical interface provided by the IEC 61850 standard between station level, bay level, and process level. The IEC 61850 standard replaces hardwiring between substation components with communication interfaces, hence simplifying substation architecture. The new substation communication architecture is enhanced by the replacement of hardwired communication cables, which decreases deployment and maintenance costs (Apostolov, 2010).

Logical Interfaces as illustrated in Figure 2.8 (Commission, 2017):

1. Protection – bay and station level data transfer between the levels.
2. Protection – bay level and remote protection data transfer between the levels
3. Bay-level data transfer
4. Bay levels current transformer and voltage transformer data transfer between levels (Sampled Value messages)
5. Control - process and bay level data transfer (measurements, status, and control) between levels (GOOSE messages)
6. Control - bay and station level data transfer between levels
7. Substation and remote engineer's workplace data transfer between levels
8. Data exchange amongst the bays (GOOSE messages)
9. Station-level data transfer
10. Control – substation device and control centre data exchange

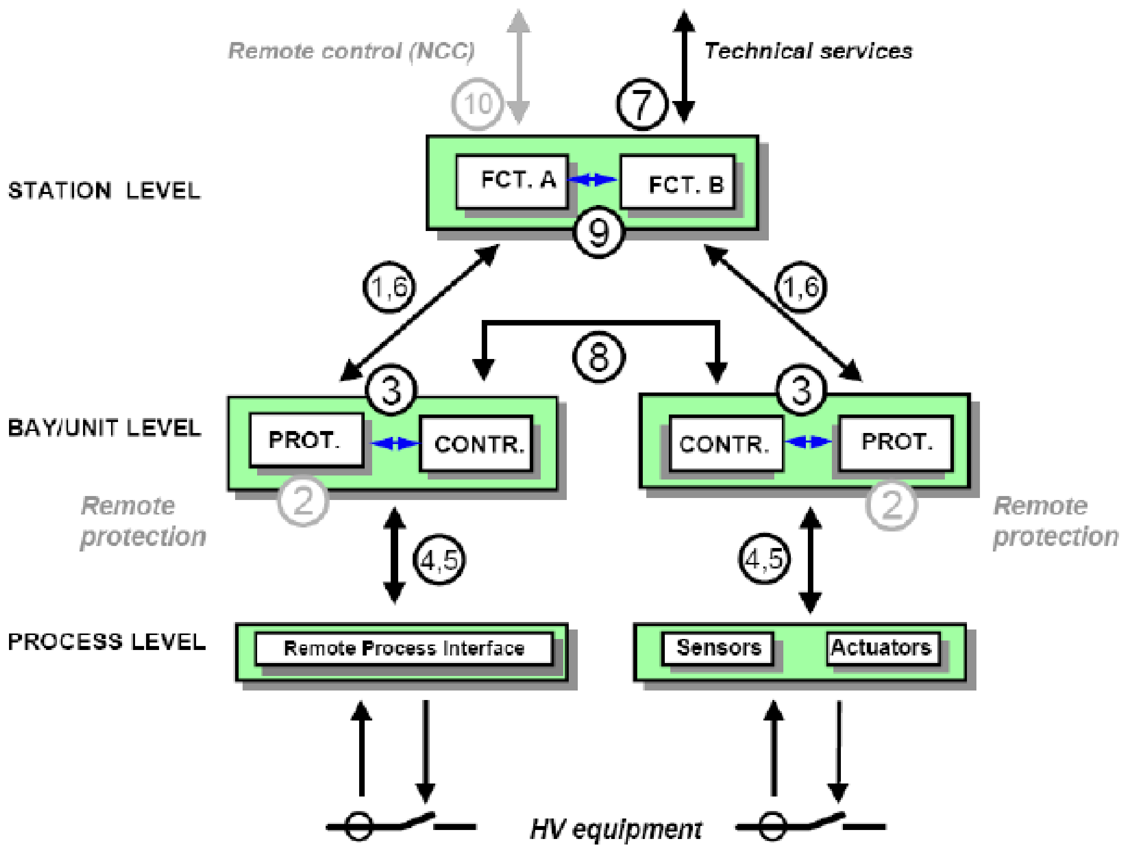


Figure 2.8: Logical Interfacing between Station, Bay, and Process Levels (Commission, 2017)

As mentioned, the emphasis switched from serial-based legacy substations to ethernet-based substations where seamless data communication is presented in the architecture between the substation levels. As depicted in Figure 2.8, the bay level comprises IEDs that communicate with the devices situated at the station level through an MMS communication service in a client-server architecture (Emmanuel, 2014). MMS is utilised for data retrieval from IEDs as well as configuration and remote access. The utilisation of the publisher/subscriber communication services allows for the establishment of inter-communication between IEDs (Emmanuel, 2014). Furthermore, the process-level devices exchange information with the bay-level IEDs using the IEC 61850 process bus communication interface to transmit GOOSE and SV messages. The information is subsequently mapped into GOOSE and SV messages using ISO 8802-3 ethernet frame (Ncube, 2012). Studying the hierarchical architecture shown in Figure 2.9; IEDs may be installed in the station, bay, and process level. IEDs installed at the station level generally serve as Human Machine Interfaces (HMIs) allowing operators to monitor and observe IED signals installed at the plant or substation. No control may be conducted via the HMI. It is the bay level IEDs that perform the control, protection, and measurement functions using GOOSE or SV messages. The process level compromises the process level IEDs which utilise the process bus to transmit information such as sampled values of voltage and current measurements.

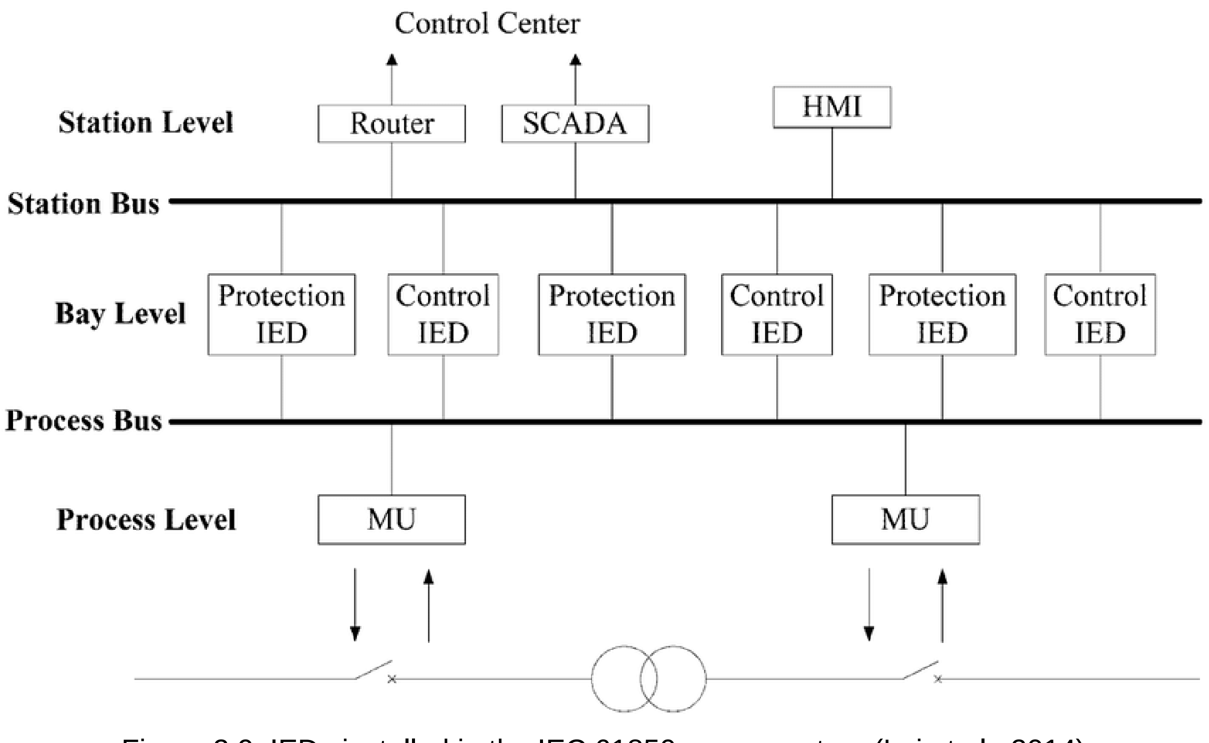


Figure 2.9: IEDs installed in the IEC 61850 power system (Lei et al., 2014)

A high-speed communication protocol between the process-level devices and the bay-level devices is provided by the IEC 61850 process bus. For the communication of GOOSE and SV messages, the process bus protocol uses ISO 8802-3 ethernet protocol. As compared to serial, hardwired communication, IEC 61850 process bus comes with benefits such as:

1. The implementation cost is reduced due to the decrease in the amount of copper cabling being used.
2. Reduce the wire resistance to prevent saturation in instrument transformers (Apostolov, 2010).
3. Improves the safety of the substation by preventing open circuits (Apostolov, 2010).
4. Elimination of legacy protocols by employing easy and accessible TCP/IP and Ethernet technology.
5. Ultimate adaptability and interoperability of the system

Based on the above discussion, IEC 61850 provides a greater advantage than legacy communication protocols in substation automation systems. Any cyber security technique that seeks to secure IEC 61850 communication messages must consider these performance class requirements. The IEC 61850 protocols are transmitted on a process bus as unencrypted ethernet packets. The IED data can be sniffed and tampered with by an attacker, thereby weakening the security. The quick response times needed for certain types of communication, combined with the restricted capacity of some IEDs to process, present a clear challenge. These challenges need to be studied and see how IEC 62351 addresses them.

2.4.2 Theoretical Framework

Cyber-physical systems (CPS) have security concerns and those challenges need to be addressed and solved. (Yoo & Shon, 2016), present a paper where they study the IEC 61850-based challenges and research directions for the heterogeneous cyber-physical system. In addition, the vulnerabilities, security specifications, and security architecture are addressed. Security issues are closely examined in the electrical grid in Korea. (Yoo & Shon, 2016), also, address the standardisation of smart grid communication protocols that correlate to the IEC 61850 standard that is based on substation automation.

The IEC 61850 standard and other smart grid protocols are presented and reviewed. The analysis of IEC 61850 standardisation and other smart grid protocols will be discussed. (Yoo & Shon, 2016), briefly discusses the IEC standards and correspondence protocols IEC 61850, DNP3, IEC 61970, and OPC UA. The original edition of the IEC 61850 standard was released in 2002, however, the information provided remained restricted to communication within a substation network (inside). The IEC 61850-7-410 version which described the object model in the hydropower environment was created in 2007, meaning the application available to the outside of the substation. IEC 61850 is designed to help model and self-describe information in an object-oriented way. As such, it is stated that the fundamental variations between IEC 61850 and other smart grid communication protocols cause problems in the harmonisation of protocols and may pose security threats.

The IEC 61850 information model can be mapped or transmitted in three ways: MMS, GOOSE, and SMV. GOOSE and SMV are transmitted over TCP/IP networks or use Ethernet for high speed. Furthermore, (Yoo & Shon, 2016) discussed briefly that the safety aspects of the IEC 61850 standard are laid down in IEC 62351. The IEC 62351 sets out how protocols based on TCP/IP can be secured via Transport Layer Security (TLS). The standard is revised for GOOSE / SMV messages based on a message authentication code being generated.

DNP3 is a communication protocol for communication transmission between the control centre and its power system. In 1993, the DNP3 protocol was developed and in 2010, it was standardised as IEEE 1815-2010. A study was conducted by the Smart Grid Interoperability Panel to standardise the DNP3 and IEC 61850 mapping process. The interoperability between the DNP3 and IEC 61850 is inevitable in this environment. The security mechanism for the DNP3 messages is specified in IEEE 1815-2010 and IEEE 1815-2012 as Secure Authentication Versions 2 and 5 (Yoo & Shon, 2016). The IEEE 1815-2012 standard was subsequently revised to strengthen Secure Authentication to provide application layer functions to verify the source of the messages, the integrity of the message, and the ability to update keys remotely using cryptography and security statistics to check events of authorisation failures.

(Yoo & Shon, 2016) further study IEC 61970, providing a structured framework for the representation of specific power system objects with object-oriented relationships for these objects; The IEC 61970 specification is employed in control centres, and together with the IEC 61850 standard, it is a primary standard for the smart grid. Many applications require the implementation of systems across the smart grid environment; thus, the IEC 62361-102 standard is being developed, which harmonises the IEC 61970 and IEC 61850 specifications.

Lastly, (Yoo & Shon, 2016) reviewed the Open Platform Communication Unified Architecture (OPC UA). The objective of OPC UA is to boost interoperability by providing a standard interface for the exchange of messages between Windows-based applications and various field devices within the industrial network environment. OPC UA provides a platform-independent environment by introducing a service-oriented architecture concept (Yoo & Shon, 2016). Therefore, the power system in Korea uses a mix of heterogeneous standards such as

IEC 61850, DNP3, IEC 61970, and OPC UA for data communication. Communication standards are interconnected between IEC 61850 and DNP3 and IEC 61850 and IEC 61970. A mapping approach has to be pre-determined to achieve conversion between different communication protocols (Yoo & Shon, 2016). When mapping to IEC 61850 and DNP3, a protocol mapping table can be created using the DNP3 XML file, the Substation Configuration Description (SCD) file, and the mapping rules as defined within the IEEE 1815 standard. Furthermore, a configuration tool can be helpful for pre-setting protocol mapping (Yoo & Shon, 2016). The link between the IEC 61850 and IEC 61970 specifications concerns data exchange services and information models standardised in the substation automation environment. However, both standards are not mutually compatible and have been designed for different purposes. The IEC 61850 standard defines the functional elements of the substation's IED as Logical Nodes (LNs). The IEC 61970 standard defines the power system resources and their relationship in the Unified Modelling Language (UML). Therefore, the interlink between IEC 61850 and IEC 61970 should be addressed in terms of both the information model and the data exchange service. Mapping between IEC 61850 and IEC 61970 information models may be impossible; a unified information model must be developed by incorporating the IEC 61850 information model with the Common Information Model (CIM).

In IEC 60870, protocol mapping refers to the process of transferring information across multiple communication protocols. The IEC 60870-5 protocol standard provides telecontrol, teleprotection, and other telecommunication operations for power systems. IEC 60870-5 is an adopted standard for telecontrol equipment and systems. Protocol mapping is especially important in the context of IEC 60870-5-104, a companion standard that specifies the usage of the TCP/IP protocol suite for telecontrol (Lin & Nadjm-Tehrani, 2018). The standard covers frame structure, information field content, and data transfer processes.

a) Application Layer - The application layer is responsible for the frame structure of the data to be transmitted. It specifies information objects and application service data units (ASDUs).
b) Transport Layer - IEC 60870-5-104 utilises the Transmission Control Protocol (TCP) for reliable data transmission. The ASDUs are incorporated into TCP packets for transmission between devices.
c) Network Layer - The Internet Protocol (IP) is used to address and route TCP packets. ASDU data is bundled into IP packets to enable end-to-end network communication between devices.

The IEC-60870-5-104 protocol is widely used in SCADA networks to manage critical infrastructure. DNP3 and IEC-60870-5-104 were both designed particularly for SCADA communication beyond the substation level; however, DNP3 has a few advantages over IEC-60870-5-104. DNP3 is capable of transmitting high data quantities over great distances while also enabling more rapid transfer of data. To benefit from the effective data transfer and execution of IEC 61850, IEC-61850 is commonly used for data mapping of IEC-60870-5-104. (Lin & Nadjm-Tehrani, 2018; Anon, 2021)

2.4.3 Application, performance evaluation, and development of IEDs capable of publishing IEC 61850-8-1 GOOSE messages

One of the communication interfaces presented in the IEC 61850 standard is the GOOSE message. The GOOSE interface facilitates communication between IEDs in different levels of a SAS. The GOOSE message maps data and object models into an ethernet protocol.

GOOSE communication is time-critical messages that are sent timeously when the occurrence of a certain event in the data and object model changes (Ncube, 2012). Because GOOSE messages are transmitted regularly, they can be used to transmit analogue and binary data to the receiving IED. GOOSE message implementation comes with the benefit of reduced wiring and engineering fees as the ethernet frame or network is used on the process bus. The process bus communication network should complete communication of GOOSE and SV messages within 4ms, as these are time-critical messages (Weerathunga, Pubudu Eroshan, 2012). The IEC 61850-5 standard specifies the time it takes for various types and classes of messages to be transmitted. According to the IEC 61850-5 standard, all devices that comply must be capable of transmitting messages of unique types or classes within the specified timing requirements. However, considering that, GOOSE communication is categorised as GOOSE trip (Type 1A) and GOOSE block messages (Type 1B). GOOSE trip messages are further classified into two performance classes, P1 and P2/P3, based on the transmission time. For P1 and P2/P3, the time requirement is 10 ms and 3 ms, respectively (Committee et al., 2017).

To verify that devices in the design stage meet the timing criteria for IEC 61850-8-1 messages, (Gonzalez-Redondo et al., 2013) explore strategies for monitoring the GOOSE communication transmission time. It is possible to measure time using three different techniques: the round-trip test, the ping-pong test, and the rally test. The IEC 61850-10 standard provides the round-trip time for calculating the time delay of a subscribed message being retransmitted back to the sending device. With the ping-pong technique, the Device Under Test (DUT) broadcasts a message to a receiving device. As soon as the message is received by the subscribing device, it sends out a GOOSE message that is then subscribed to by the DUT. Lastly, a rally technique is conducted, in which both devices simultaneously issue GOOSE communication, which is subsequently received by the other device. After the device has received a message, it will broadcast a new message, which the other device will then subscribe to. As a result, both devices will be constantly energised. According to the configuration, a client-server architecture is implemented. Using this practical configuration, the findings indicate that the greatest round time trip time for a GOOSE message is 1.536 ms, which is less than the time requirement of 3ms (Gonzalez-Redondo et al., 2013).

Figure 2.10 details the communication stack of the protocol. The fast transmission time of messages is guaranteed as GOOSE messages are mapped to the second layer of the stack interface (Hohlbaum et al., 2010). The SV and GOOSE protocols map straight into the ethernet frame. As specified, the MMS protocol can work via TCP/IP or ISO. Using "Ether-type" or "802.3", all data is mapped into an Ethernet layer.

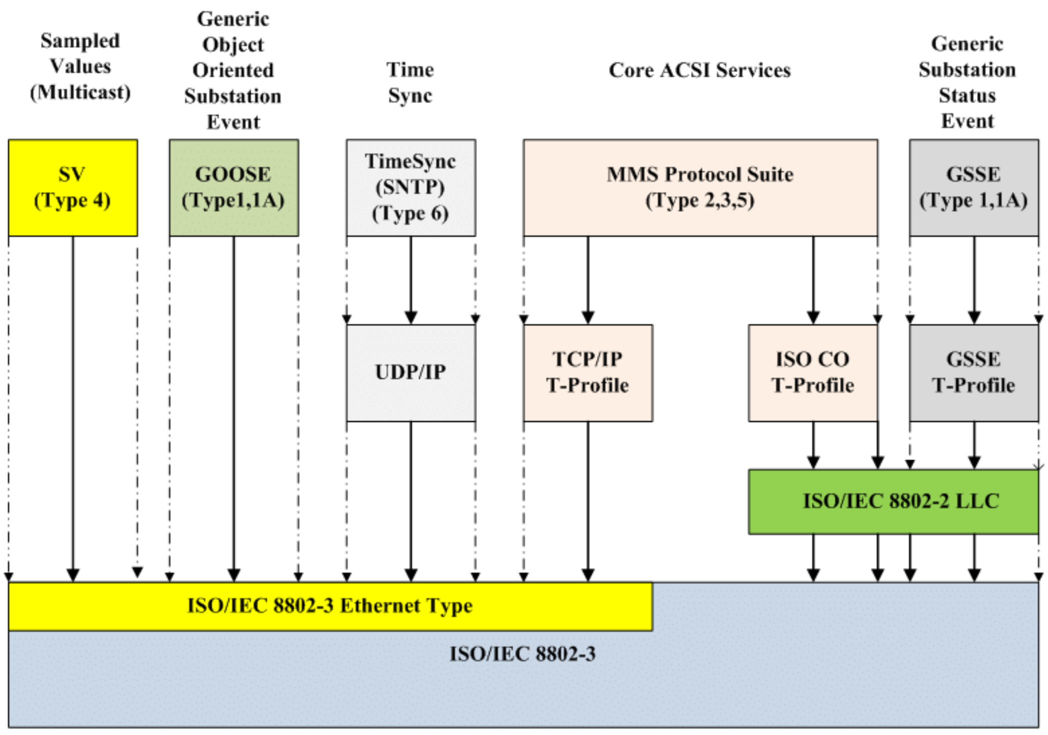


Figure 2.10: Message communications stack in IEC 61850 (Ali et al., 2016)

As previously discussed, GOOSE communication can be utilised to transmit data between IEDs or between process equipment such as transformers or circuit breakers. Analogue values and or digital signals can be transmitted via GOOSE messaging to IEDs located on the bay level.

GOOSE messages offer numerous advantages over conventional communication protocols. The benefits of using GOOSE are listed below (Ncube, 2012; Emmanuel, 2014):

1. High-speed transmission between substation IEDs.
2. Reduced point-to-point copper wiring for protection and control networks. (Daboul et al, 2015) examine the performance and timing of GOOSE messages when used in conjunction with serial communication in a SAS. Two IEDs were designed to exchange GOOSE messages via serial communication; the findings indicate that the GOOSE round-trip duration is roughly 2.5 ms, but the hardwired signals exchange takes 20 ms. This article demonstrates that GOOSE messaging can be used to exchange information in SAS without impacting the system's operation.
3. The latest GOOSE messages are regularly available on the process bus as they are transmitted as a multicast message; as such providing high system availability.
4. The GOOSE protocol does not support message acknowledgements. Hence the amount of traffic in the process bus is reduced guaranteeing high delivery rates.
5. Flexibility and expandability for protection and control schemes.
6. GOOSE capability to be utilised beyond substations and on different smart grid applications through Routable GOOSE (R-GOOSE).

GOOSE protocols are adaptable for implementation in a wide range of applications. Table 2.4 shows a catalogue of articles that describe the use of the GOOSE communication protocol in an IEC 61850 standard-based SAS.

Table 2.4: IEC 61850 GOOSE Literature Review

Paper	Research Objectives	Method	Outcomes
(Apostolov & Vandiver, 2011)	The authors study the IEC 61850 GOOSE applications to protection systems.	The authors study the need for improved electric supply by investigating failures in the distribution protection scheme. Furthermore, a simulation tool is proposed to test the system.	The use of IEC 61850 GOOSE messages significantly improves distribution substation protection by reducing fault clearing times and mitigating the effect of short circuit faults. By utilising high-speed messaging, many hardwired connections are eliminated.

(Gonzalez-Redondo et al., 2013)	The IEC 61850 GOOSE strict time requirements are studied. This document includes a brief description of various performance tests.	A simple testbed for evaluation purposes is presented. A testbed for doing a first performance evaluation of a Commercial Off The Shelf (COTS) solution for IEC 61850 devices was used to demonstrate a straightforward methodology based on physical transfer time measurements.	Tests on the GOOSE message demonstrate that the system is appropriate for the most demanding applications with a transfer time of approximately 1.5 ms.
(Omar Hegazi , Eman Hammad , Abdallah Farraj, 2017)	This article describes the modelling and construction of a GOOSE traffic generator in the Riverbed Modeler. We quantify the extra processing latency introduced by the traffic generator's implementation in terms of encoding/decoding.	The generated generator was incorporated into the Riverbed Modeler, which required the development of a customised model to support IEC-61850 in Modeler.	The simulation results demonstrated the customised models' low overhead. The primary contributions of this work are a full modelling of the GOOSE message and a message generator for extracting status and value data. The proposed message generator can be used to convey time-sensitive GOOSE signals quickly and reliably, as well as expand research on cyber-security.
(Harispuru & Schuster, n.d.)	This article demonstrates how to configure and test GOOSE communication to achieve maximum dependability without losing performance.	The purpose of this study is to discuss the tools and approaches for increasing the reliability of communication at the IED level via GOOSE messaging.	Numerous diagnostic methods for GOOSE signals were described. Test programs can be used to validate the network's data traffic. These test programs can verify that GOOSE messages are being transmitted correctly across the network and that they match the preset data.

(Elbez et al., 2018)	The document describes the design and implementation of a software testbed to study the CPS of IEC 61850-based electrical substations.	The suggested testbed provides an appropriate level of detail/abstraction for both the network and physical components, as well as their communication. The proposed testbed is unique in that it integrates the physical and cyber components of the power system.	The current testbed was developed specifically to analyse the cyber-physical security of electrical substations per IEC 61850. Additionally, the proposed testbed will be utilised to demonstrate defence mechanisms for securing the network connection of modern electrical substations.
(Fernandes et al., 2014)	The paper provides an overview of the GOOSE protocol and its benefits.	In this study, the authors presented a demonstration of the use of GOOSE capabilities in a protection IED to demonstrate its functionality. The results of two such IEDs being tested in a laboratory environment are discussed.	The IEC61850 GOOSE connection between the two devices was successfully established, with both devices able to publish and receive GOOSE messages. The seamless integration of IEC61850 into the IED simplifies the process of developing power system protection strategies.
(Bhamare, n.d.)	This article discusses how to implement several of the new capabilities included in IEC 61850, Communication Networks and Systems in Substations.	The article discusses how to leverage GOOSE communication to increase the system's dependability and performance between protection and control devices. The author further studies the network architecture for GOOSE communication to better reduce hardwiring and provide a more modern and reliable protection and control system.	IEC 61850 is a good investment because of the cost reductions associated with substation design, installation, commissioning, and operation, as well as the addition of new capabilities that are not feasible or cost-viable using traditional methodologies. Utilising IEC 61850 and GOOSE significantly improves the performance and speed. Additionally, the

			usage of IEC 61850 and GOOSE increases the operational reliability of the protection system.
(Pathan & Asad, 2016)	This article discusses in depth the GOOSE-based ACSE scheme logics for power transformers, as well as their latency.	The scheme is optimised for performance using the software PCM V2.5, which eliminates the need for redesign and rewiring. The simulation results indicated the latencies of GOOSE messages on a system setup for an ACSE scheme.	Automatic changeover switching equipment (ACSE) transfer systems improve the reliability of a power system by automatically switching to a standby power source when the primary source experiences a power outage. The authors conclude that the proposed approach is reliable since old wiring can break without warning.

These previous research studies demonstrate that GOOSE messages can be used to communicate time-sensitive communications over a low-traffic Ethernet network. As part of the implementation of an electrical fault protection scheme that uses GOOSE, the system engineer must ensure that, in the event of an electrical fault, the protection scheme is operational within a reasonable amount of time. When a fault occurs, there are several different methods for determining how quickly a protective system responds. The speed and performance of an IEC 61850 protection application are used to determine whether or not a system is a suitable solution. Methods for determining the precise GOOSE transfer time have also been developed. It is critical to ensure that the GOOSE transfer time is within the allowed time frame because this has a direct impact on the overall response time (Retonda-Modiya, 2012).

2.4.4 Application, performance evaluation, and development of IEDs capable of publishing IEC 61850-9-2 SV messages

As the IED continues to develop, more and more digital systems are being incorporated into the power system, and the conventional protection system is being phased out in favour of the digital system. The testing of traditional protection systems has already been completed, while the evaluation of digital systems in the power industry is still in the process of being completed.

(Engler et al., 2004) conducted several feasibility tests. The tests involved measuring the delay time of SV across an Ethernet network, as well as the performance of the IEDs when it is loaded with an SV message. During the evaluation, the accuracy of synchronisation was within 20µs. When using SV, the maximum delay time is 3.520ms. (Engler et al., 2004) confirms that the real-time requirements are conformed to the substation automation environment. (Engler et al., 2004) further dictates that modern ethernet equipment should be used to accomplish reliable operation. (Kanabar & Sidhu, 2011) further simulates a process bus for a substation using ethernet communication. Communication delays for time-critical packets (such as process bus communication, which includes GOOSE and SV) are permitted according to IEC 61850, with a maximum of 3ms to 4ms. (Kanabar & Sidhu, 2011), using fibre optic communication confirms that the delay time of SV is within the required range. The benefit of employing ethernet-based communication is that it enables the creation of more advanced communication architectures and integration with the rapidly increasing ethernet-based communication technologies. Due to the time-critical and high-speed characteristics of today's Ethernet-based communication technology, it is an ideal communication technology for automation applications in substations (Abdolkhalig, 2014). Ethernet-based communication has been proposed for use in substation automation in parts 8 and 9 of the IEC 61850 standard, respectively, at the station and process levels.

The communication network within the substation is organised into three levels: process, bay, and station. The process is at the lowest level of the network. The switchgear equipment, actuators, and sensors are all considered to be part of the process level. The process bus is the Ethernet interface between the IEDs at the process and bay level. Only a few practical implementations have used the IEC 61850-9-2 process bus standard. Incorporating the process bus has numerous clear advantages. Comparing the new approach to the previous approach, there will be a significant reduction in the amount of copper wiring required to connect the process equipment. Aside from that, automated testing is possible with a digitalised information system, which makes the installation process simpler (Zhao, 2012).

(Adewole & Tzoneva, 2014a) investigate the effect of the SV process bus on the operational performance of protective relays. The speed, dependability, and security characteristics of the sampled value process bus under investigation were among the considerations. (Adewole & Tzoneva, 2014a) compared the performance of two IEDs in a distance protection scheme by employing a hardware-in the loop-solution to a standard hardwired arrangement to see which was more effective. A Real-Time Digital Simulator (RTDS), an IED, a GPS clock, and an industrial network switch were used to configure the hardware-in-the-loop SV protection method. To evaluate the protection system's performance, it was subjected to a variety of fault locations, fault resistance, and fault inceptions with varying Source Impedance Ratios (SIR).

According to (Adewole & Tzoneva, 2014a) to evaluate the security and reliability features of the IEC 61850-9-2 standard process bus communication network, the delay was introduced into the network. Due to the similar response and tripping times between the two communication modes, the findings of laboratory testing suggest that the IEC 61850-9-2 standard process bus can be used in place of conventional hardwired communication between protection IEDs and instrument transformers. However, when compared to hardwired systems, the SV process bus offers several advantages, including the removal of parallel copper wires between instrument transformers and IEDs, as sample values are published onto an Ethernet network. (Adewole & Tzoneva, 2014a) conducted a test to demonstrate that the IEC 61850 process bus may be utilised in place of conventional protection schemes without compromising the security and operation of the system.

Merging Units that comply with the IEC 61850-9-1 standard and the IEC 61850-9-2 standard communicate SV messages to bay-level IEDs using the serial link and the Ethernet process bus, respectively, according to the standards. Each of these SV messages contains instantaneous voltage and current samples of the power system, which are sampled by the Merging Unit (MU) at a predetermined pace. The MU is a device that allows information to be exchanged between the electronic current transformer and the bay-level equipment. It is used for metering and for implementing other protective functions to use the sampled value messages received by the bay-level IEDs instead of the full value messages. To receive these published SV messages, bay-level IEDs must subscribe to them using the abstract communication services provided in the IEC 61850-7-2 and IEC 61850-9-2 standards (Ncube, 2012).

In sensor systems, such as CTs, VTs, or digital input or digital output sharing, sampled measured value (SMV) is a mechanism that is used to transport measured samples from sensor systems between IED devices. The MU depicted in Figure 2.11 accepts analogue input signals from current and voltage transformers via an Analogue to Digital Converter (ADC) and binary input signals from primary plant equipment through a digital converter. In this case, the information is mapped onto an IEC 61850-9-1 standard SV frame and then broadcast to bay controllers through a serial unidirectional multidrop point-to-point communication link. Thus, a single ethernet-based process bus network may carry a large number of digital signals.

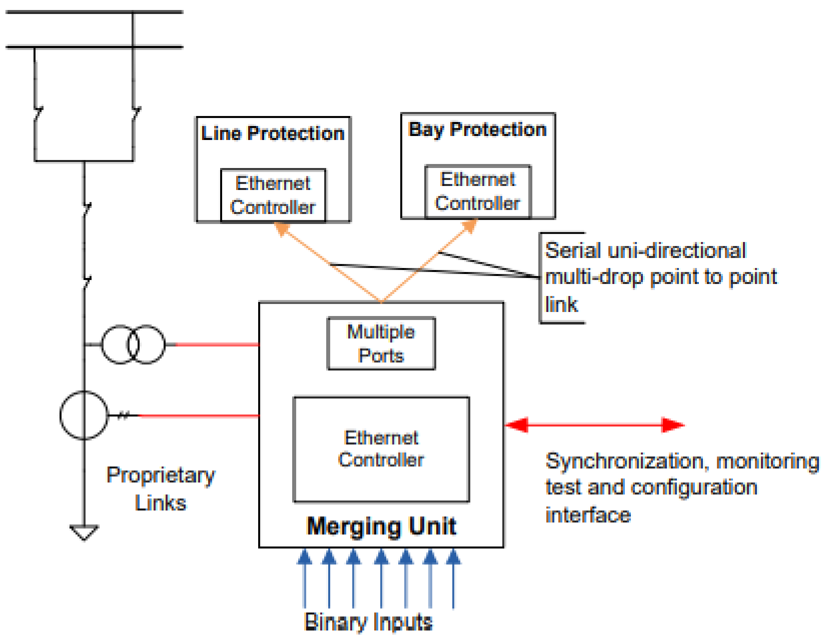


Figure 2.11: IEC 61850-9-1 standard-based Merging Unit (Ncube, 2012)

MUs are depicted in Figure 2.11 as devices that are capable of broadcasting sampled value messages carrying digital information obtained by reading digital inputs per the IEC 61850-9-1 standard (Ncube, 2012). Aside from the fact that they must continuously supply information to protection, monitoring, and control devices, sampled values are transmitted at a consistent rate, as opposed to GOOSE messages. As a result, the sampled values consume a significant amount of network traffic. Because the SMV and GOOSE messages will be the primary constraints on network capabilities, it is critical to conduct a thorough examination of these two protocols under their most extreme operation to ensure system stability.

Table 2.5 provides a literature review of IEC 61850 SV messages for use in substation automation systems.

Table 2.5: IEC 61850 SV Literature Review

Paper	Research Objectives	Method	Outcomes
(Apostolov, 2010)	The study begins by defining the Process Bus concept as defined in IEC 61850 9-2 and then focuses on the implementation that provides interoperability between merging units	In comparison to conventional protection and recording systems, applications based on the IEC 61850 provide several significant benefits. The study discusses functional	Improved system flexibility, reduced CT saturation problems, and prevention of open current circuit circumstances are only a few of the major benefits addressed in

	and protection devices. The Merging Unit's components, performance requirements, and temporal synchronisation are discussed.	enhancements in conjunction with the realistic elimination of performance or safety concerns.	the study. There are numerous benefits compared to traditional protection and control systems: Cost savings on wiring, installation, and commissioning, and simplicity of adjustment to different configurations.
(Adewole & Tzoneva, 2014b)	The objective of the research is to examine the movement toward the usage of the IEC 61850-9-2 Process Bus in substations. The impact of the IEC 61850-9-2 standard on the reliability, security, and operational speed of protective IEDs is discussed.	A lab-scale hardware-in-the-loop experiment is implemented and used for the investigations. It includes the RTDS, an IEC 61850-9-2 protection IED with SV inputs, a conventional hardwired protection IED, a GPS satellite clock, and industrial network switches. The studies are designed to compare the performance of the two protection IEDs' distance protection functions while exposed to a variety of fault types at a variety of fault locations. Additionally, the effect of random noise/delay on the protection functionalities of an IEC 61850-9-2 Process Bus-based protection IED is examined. It demonstrated that it does not affect the Process Bus IED unless security and reliability are compromised.	The results of the numerous tests conducted revealed that both IEDs operate similarly in terms of operating time responsiveness and tripping times throughout all protective zones. Additionally, the IED's dependability and security were confirmed. Integration of Process Bus-based IED improves substation safety, reduces copper usage, and simplifies maintenance and reconfiguration.

(Skendzic et al., 2007)	The IEC 61850-9-2 standard introduces the SV Process Bus idea. This standard proposes that the outputs of Current and Voltage Transformers that are now hardwired to various equipment be converted to digital signals and transmitted via an Ethernet network.	This article discusses the Merging Unit Concept, examines the reliability of the protection system in a process bus context, and provides an alternate way for successfully deploying this technology. SV process bus has several additional issues that must be handled, as well as interoperability difficulties.	A large reduction in the amount of low-voltage cable and its replacement with logical connections established through the process bus LAN are two key benefits of the proposed solution, which promises to minimise installation costs. As a result, a fresh approach is required. It must be adaptable, dependable, and capable of delivering benefits to its users.
(Ingram, Schaub, Taylor, et al., 2013)	The objective of this article is to examine the role of a communication network that is developed on the IEC 61850 standard. The information gathered is used to acquire a better knowledge of the properties of the process bus. The concept of coherent transmission, as well as its implications for Ethernet switches, are addressed in detail.	The behaviour of Ethernet switches with sampled value traffic is investigated in detail using experiments based on substation observations. The authors provide test methods for determining a network's adequacy.	Once an Ethernet switch queue sampled value frames, future switches incur minimal additional delay, as such communications cabling can be cut without impairing operation. A process bus network's performance and reliability have been proven to be satisfactory.
(Engler et al., 2004)	This document provides an overview of the feasibility studies completed with IEC 61850.	The IEC 61850 standard makes extensive use of industry-standard protocols like TCP/IP and switched Ethernet. The advantage of this approach is that it enables the employment of readily available and acceptable communication	The feasibility studies indicate that the communication solution presented by IEC 61850 satisfies the control and protection specifications.

		components. While the use of Ethernet for connection between stations and at the bay level looks to be trouble-free, it is vital to confirm that the same technology can be utilised for time-critical communication.	
(Hodder et al., 2009)	This paper discusses the primary motivations and expectations for next-generation protection and control systems across today's utilities. A practical design is described that provides both strong technical performance and significant cost reductions to the user, with cost savings and resource optimisation demonstrated through a simplified business case study.	A fundamental business justification for building IEC 61850-9-2 process bus architectures must be to minimise all costs related to the installation, operation, and maintenance of protection and control systems, as well as to optimise project execution and resource use.	Recognising the current cost structure, a technical solution for protection, control, and automation has been designed that is both simple and practical, with a high potential for quick adoption. The architecture is focused on delivering a system that complies with IEC 61850, is cyber secure, and provides business benefits.
(Starck et al., 2013)	This article describes how to optimise switchgear using IEC 61850-9-2. The paper further describes how sampled values can be used to improve the reliability and functioning of a medium-voltage substation's IEDs.	This article describes a way to combine station and process buses into a common bus, complete with an example and availability analysis. Calculated findings suggest that availability, performance, and reliability have been improved.	In comparison to typical instrument transformers, the use of non-conventional instrument transformers in conjunction with IEC 61850 real-time communication provides more cost-effective solutions with increased availability.
(Kumar et al., 2016)	The purpose of this article is to analyse the efficacy of a smart protection system in a zone substation that	This article provides simulation results for packet transit delay in an Ethernet environment, which may	In conclusion, while old copper wires linking relays in a conventional substation functioned well at the bay and

	utilises IEC 61850-9-2 relays.	have a significant influence on the SAS network's protection system. Increasing the SV's frequency resulted in increased packet losses per second during the performance simulation.	process level, it is recommended to consider the deployment of the IEC 61850-9-2 process bus design to save costs and accelerate project implementation time. The ease of wiring and flexibility of changing devices without having to shut down the secondary system are the main advantages of the process bus design.
(Gurusinghe et al., 2018)	The purpose of this study is to demonstrate a methodology for testing a fully digital SAS utilising a real-time power system simulator.	A real-time simulator, four multi-functional protection IEDs, a MU, an amplifier, a GPS clock, a GPS antenna, two Ethernet networks, and a workstation to execute the essential software are included in the suggested test setup. Such testing requires a complex testing infrastructure and a significant amount of engineering work.	IEC 61850 SV enables interoperability, as well as cost savings associated with commissioning and maintenance. SV enhances the safety and dependability of the substation environment.

Ethernet process bus communication networks that adhere to IEC 61850-9-2 specifications can provide numerous advantages and benefits. The advantages of ethernet switched communication, which is specified in IEC 61850-9-2, such as the data speeds, zero-collision, and flexible design, make it preferable to the serial point-to-point standard links standardised in IEC 61850-9-1 (Abdolkhalig, 2014). The process bus, which is based on an Ethernet communication network, was proposed by the IEC 61850 standard to reduce the expense of engineering and wiring of long copper wires between the process level and the bay level or control room in the substation. Other than cost savings, the process bus has an architecture that is simple, flexible, and interoperable, among other characteristics. The IEC 61850-based process bus, in particular, has several technical difficulties. The latency and loss of time-critical SMV messages via the ethernet process bus communication network are the most significant technical challenges that have been explored by (Engler et al., 2004, Kanabar & Sidhu, 2011, Adewole & Tzoneva, 2014, Abdolkhalig, 2014). To successfully implement a protection application in a substation the communication availability and reliability must be investigated as well. The IEC 61850-8-1 proposed that the same GOOSE message be sent numerous times to improve transmission reliability. However, because SV messages are not repeated during transmission, the reliability of their transmission is reduced. As a result, it is critical to analyse and research the dynamic behaviour of process bus communication based on IEC 618509-2. The impact of SV loss or delay on substation phasor estimation and digital protection should be mitigated through the development of appropriate approaches for any future digital protection and automation systems. Other considerations for the development of the process bus include time synchronisation and data security. etc., (Abdolkhalig, 2014, Kasztenny et al., 2005).

Precision Time Protocol (PTP) and time synchronisation are pivotal components of power utility automation for ensuring precise and synchronised time across multiple devices and systems. Within the framework of IEC 61850, specifically Part 9-3, there is a dedicated focus on addressing time synchronisation. This standard describes the utilisation of PTP to synchronise clocks, ensuring interoperability across various devices. PTP enables highly accurate time synchronisation by facilitating the exchange of timing information among networked devices. Numerous power utility applications, such as energy management, fault logging, and protective relaying, depend on precise time synchronisation. PTP guarantees that electrical grid devices operate at exact times, improving system efficiency and reliability. The successful implementation of PTP must take into consideration factors such as network latency, delay variation, and device clock accuracy.

(Lin & Nadjm-Tehrani, 2018) investigated the application of Precision Time Protocol (PTP) for synchronising the sampled value Process Bus. The study involved comprehensive tests to assess the performance of PTP within the Process Bus across varying Ethernet network loading scenarios. The experimental setup comprised a network switch and a Global Positioning System (GPS) utilised to emulate merging units (MUs) for imposing network loading conditions. An external time reference device was incorporated to compare the synchronisation accuracy between master and slave clocks under diverse loading conditions. The experimental results indicate that Process Bus traffic has the potential to disrupt PTP synchronisation due to packet losses, leading to a delay exceeding 1μs in the slave clock relative to the master clock. Furthermore, the experiments highlight that outcomes are achieved when there are no merging units (MUs) traffic or when Virtual Local Area Networks

(VLANs) are employed, as opposed to sharing the same network for Sampled Value (SV) and PTP communication.

(Ingram, Schaub, Campbell, et al., 2013) delved into an examination of the effectiveness of Precision Time Protocol (PTP) components within the context of IEC 61850 sampled value Process Bus systems. The methodology outlined in this study encompasses a diverse range of tests that can be employed by system designers for the thorough analysis of timing components. The authors put forth a synchronisation system that aligns with the comprehensive functional requirements of the system. As per the findings presented by the authors, PTP devices engineered for deployment in power systems demonstrate interoperability, ensuring accurate synchronisation for each grandmaster and slave clock pair. However, it is imperative to acknowledge the substantial impact of clocks employed within a substation timing system on its overall performance. The study reveals that the implemented PTP system successfully adheres to the ±1 μs standards outlined in IEC 61850-9-2 Light Edition (LE), even when utilising a shared process bus network for both SV communication and time synchronisation.

(Shrestha et al., 2021) conducted a comprehensive investigation into the consequences of time synchronisation and network-related issues on protection within digital secondary systems (DSS). In the context of an IEC 61850-based DSS, synchronisation and SV data exchange occur via an Ethernet network. Optimal functionality of protection functions mandates the maintenance of both time sources and an efficient protection network. The authors present a test case illustrating the impacts of disabling protection function operation during instances of network congestion. Addressing time synchronisation challenges, potential issues arise from non-functional GPS signals, uncertain dependability and redundancy of satellite clocks, and different synchronisation sequences among MUs and protective relays from different manufacturers. Likewise, inadequately configured Ethernet networks may lead to packet loss or substantial network delays, causing temporary incapacitation of protection functions or degradation in overall protection speed. The study advocates for the consolidation of communications and time synchronisation services over a unified channel within an IEC 61850-based DSS. This approach ensures precise time alignment among all communicating devices, with clearly defined failure and recovery processes. The overarching objective is to guarantee synchronised operation among all communicative devices within the DSS, emphasising that proper functionality is dependent upon accurate time synchronisation. Loss of synchronisation within a DSS induces an artificial phase shift, potentially leading to erroneous tripping and detrimental effects on protective function availability.

The following section conducts a review of literature on the IEC 62351 standard in detail focusing on available security techniques that can be employed.

2.5 Literature Review – IEC 62351

2.5.1 Introduction

As the power system becomes more advanced, with the integration of IoT and IT, cyber security becomes a feature to be considered as the development of technology poses new security risks. The operation that is managed between IEDs that are connected by communication networks has a high transmission of information, and the cyber system has been widely implemented for monitoring, controlling, and protecting, which is why there is a requirement for security. Data and communication in the power system are to be securely protected to maintain a safe electricity supply as the energy sector is a critical infrastructure. However, it must be noted that *"no industry can eradicate risk entirely when determining security strategies against current threats facing the energy sector"* (Karnati, 2020). Implementation of a security standard ensures compliance in performance and interoperability. Cyber security is important for its critical infrastructure. Integration of cyber security policies and measures from the planning of a communication network to its deployment and maintenance are essential to make the network resilient to external attacks and internal carelessness. The IEC 62351 standard addresses security challenges and countermeasures in power system communication networks, while the IEC 61850 standard does not incorporate security elements (Karnati, 2020). Understandably so, cybersecurity was not an issue when the IEC 61850 standard was first published.

IEC Working Group (WG) 15 of Technical Committee (TC) 57 published IEC 62351 on security for IEC 61850 because there had been insufficient progress made in the direction of incorporating security within the communication standard (Nozomi et al., 2019). IEC 62351 is a cyber security standard (accepted internationally) that is focused on improving and delivering security to the power system and further developing cyber security measures for GOOSE and SV communication. The standard seeks to ensure the credibility, reliability, and confidentiality of the different protocols used in the automation of substations. Cyber security is defined as a method for securing the transmission of data on a communications network. Det Norske Veritas Germanischer Lloyd (DNV GL) is one of the companies that offer IEC 62351 verification and conformance testing. These procedures help prove the security level and interoperability of the power system including solving compliance issues. Rigorous testing is essential to ensure that the infrastructure is secure.

With the 4th Industrial Revolution (4IR), automation has become of paramount importance in life. Industrial control automation is available in different industries and as such manages electricity, water, transportation, power stations, etc. These critical infrastructures must be protected from cyber-attacks, failure, and damage or this may lead to economic destruction (Schlegel et al., 2017b). Improving security against targeted cyber-attacks and hacker virus attacks is of paramount importance. Therefore, IEC 62351 addresses the security of substations and protocols used in power systems. (Schlegel et al., 2017b) puts it that, "*like many of these standards, it is not a revolution, but a careful evolution, to address security issues without completely breaking backward- compatibility and interoperability with legacy systems*". The standard is made up of ten different parts, addressing different areas. Table 2.6 provides a summary of the major sections of the standard.

Table 2.6: Summary of IEC 62351 Standard (Cleveland, 2012)

IEC 62351 Parts	Title
Part 1	Communication Network and System Security Introduction to Security Issues: An overview of the standard is provided and aims are highlighted. It also provides information on security and security attacks.
Part 2	Glossary of Terms
Part 3	Communication Network and System Security Profiles including TCP/IP: This standard focuses on TCP/IP-based security protocols. The aim is to ensure the authenticity and integrity of transport layer data, and preferably also confidentiality using encryption mechanisms.
Part 4	Profiles including MMS: This section addresses security protection for profiles such as the MMS.
Part 5	Security for IEC 60870-5 and Derivatives: The fifth section, defines security protection for IEC 60870-5 protocols, as well as derivatives such as DNP-3.

Part 6	Security for IEC 61850: This section of the standard discusses security protection for protocols as defined in the related IEC 61850 Standard. This section suggests an extension for GOOSE and SV messages Protocol Data Units (PDUs) of IEC 61850.
Part 7	Objects for Network Management: This part of the standard describes the types of data structures to be used which are unique to power systems.
Part 8	Role-based Access Control: This section describes system-wide role-based access control for the infrastructure of power systems. It addresses different access types, such as direct and remote access.
Part 9	Key Management: This section intends to address key management.
Part 10	Security Architecture Guidelines: This IEC 62351 section provides general guidance on the security architecture of power systems. This offers a description of the security controls that can be introduced in the power systems, as well as interface design guidelines on how to develop communication infrastructure for power systems.

IEC 62351 parts 3, part 4, part 6, and part 10 are significant for the research, and as such an assessment is provided in Table 2.7. There are other available cybersecurity standards such as the IEC 62443 standard and National Institute of Standards and Technology (NIST) cybersecurity framework.

Cybersecurity constitutes a fundamental element within Industrial Control Systems (ICS). The IEC 62443 series of standards presents a pragmatic and attainable framework for addressing security risks. Its primary aim is to diminish the vulnerabilities inherent in the establishment and operation of Industrial Automation Control Systems (IACS). Familiarity with the structural organisation of IEC 62443 and the associated obligations is conducive to upholding robust cybersecurity within IACS, which are integral components of smart grids.

IEC 62443 prescribes that IT security professionals typically conduct network and device scans as part of routine vulnerability assessments and security evaluations, as outlined in IEC 62443-3-1. Additionally, suppliers of software patches are required to conduct vulnerability assessments in accordance with IEC 62443-2-3. Furthermore, all components of smart grid architecture must undergo compliance testing against the set of standards. (Dolezilek et al., 2020) emphasized the utilization of Industrial Control Systems (ICS) methods and standards to develop defense-in-depth cybersecurity measures for digital communications within an Energy Control System (ECS) network. The ECS communications architecture, being mission-critical, is structured into various levels, each with distinct requirements and features ranging from the process level to the control centre. By leveraging these levels, the authors advocate for the identification of interconnected cyber defense technologies, determining their deployment levels, and associating them with specific devices, as outlined in IEC 62443 Part 3. This approach contrasts with the broad defense-in-depth strategy of mandating all devices to incorporate every cyber defense technology, as suggested by IEC 62443 Part 4. (Dolezilek et al., 2020) underscores the significance of designing system security rather than relying solely on device security. They argue that adopting a defense-in-depth strategy ensures comprehensive cybersecurity, unlike the insufficient device-level security measures outlined in IEC 62351 and IEC 62443 Part 4. Ultimately, they assert that defense in depth is the appropriate method for effectively securing modern ICSs against both malicious and non-malicious cyber threats.

The National Institute of Standards and Technology Special Publications (NIST SP) 800-82, Guide to Industrial Control Systems (ICS) Security, comprehensively addresses the security of Industrial Control Systems (ICS). NIST SP 800-82 introduces security controls pertaining to security assessments, including Security Assessment and Authorization procedures, which validate the proper application and functionality of specific security controls, ensuring the desired outcomes. Recommendations regarding penetration and vulnerability testing tools are provided, mindful of instances where the utilization of such tools resulted in disruptions to ICS operations due to increased traffic and exploits. Thus, it is advisable to meticulously evaluate the potential impact of these tools on ICS operations beforehand.

Table 2.7: Assessment for Significant IEC 62351 Parts(Schlegel et al., 2017b)

Part	Title
Part 3	This part aims to implement message integrity protection and confidentiality. The standard IEC 62351-3 addresses the need to utilise Transport Layer Security (TLS) and X.509 certificates to provide encryption and authentication. Utilising TLS for substation automation is a suitable and appropriate choice that provides a degree of security. The standard, however, allows the use of NULL ciphers which do not use encryption. This is risking incompatible implementation and hence some

	degree of sufficient security is not provided. Another major issue with this part of the standard is backward compatibility reducing the security provided.
Part 4	The standard provides security for application-level security profiles and for TCP / IP-based profiles. However, differing security mechanisms will be achieved, depending on whether encryption is used or not. If encryption is used, part 3 of IEC 62351 must be applied to achieve authentication, integrity, and 69iality, if not unauthorised access to information. The main problem is that if IEC 62351-3 is not used with the standard, then no integrity or confidentiality of the message will be provided, only initial authentication. The authentication also has a time stamp, which must be correct. If not accurate, the security profiles at the application level are open to at least three attacks.
Part 6	Part 6 is an appropriate and suitable standard. However, there is no provision for time-critical traffic. Many applications need a response time of 4ms, and IEC 62351-6 does not recommend encryption for such applications. The use of RSA signatures is recommended for the authenticity and integrity of extended PDUs. For applications requiring a 4ms response time hash-based message authentication code (HMAC) can be implemented.
Part 10	Part 10 sets out comprehensive and precise architectural and security control details.

2.5.2 Methodology Implementation in Literature

(Hohlbaum et al., 2010) practically considered the performance evaluation of IEC 62351-6 and concluded that the potential software (cryptography algorithms) and hardware solutions could not meet the necessary performance requirements for GOOSE and SV data. As such, literature will be studied to address symmetric cryptography. (Schlegel et al., 2017b) approves that the IEC 62351 standard can meaningfully improve security in substation automation and control systems if applied correctly: providing authentication, integrity, and confidentiality of data. However, backward compatibility remains a concern because there is some extent of security loophole. Various mechanisms are paramount to improving the security architecture of power systems. (Tesfay & Le Boudec, 2018) investigate better performance algorithms such as the Elliptic Curve Digital Signature Algorithm (ECDSA). The authors proved that the latter algorithm has features of being faster and requiring lower computational power than RSA. (Farooq; et al., 2019) further studies various security algorithms for IEC 61850-based messages to secure GOOSE, Routable-GOOSE, SV, Routable-SV, and MMS. GOOSE and SV messages sent via IP networks are referred to as “routable” as per IEC 61850-90-5. In essence, R-GOOSE and R-SV implement the Application Profile and employ routable UDP to permit data transfer across WAN. Notably,(Karnati, 2020) observes a loop when it comes to securing SV messages as more research needs to be conducted. (Karnati, 2020) designs implement, and further conducts a performance analysis of IEC 62351 for providing security mechanisms for SV by examining the structure of the SV communication protocol. The platform is tested and validated using Hash-based Message Authentication Code (HMAC) and Galois MAC (GMAC). IEC 61850 uses ethernet and TCP / IP and therefore firewalls can protect the perimeter of security and VPN technology can build secure channels to remote centres. Access to the systems must be secured along with comprehensive logging of all user accounts by user authentication and authorisation. As follows, Figure 2.12 depicts potential security attacks in the substation automation environment:

1. A1; a cyber intruder may compromise the user interface (SCADA, HMI). Access control measures are to be put in place for identification and authentication to prevent unauthorised access. The remote access points must have suitable security frames implemented.
2. A2; cyber-attack can disrupt time synchronisation and operators will lose communication with the power system.
3. A3; compromise station bus.
4. A4; an attacker can gain access to the bay-level devices.

5. A5; an intruder can alter the device protection settings by employing a Protection Setting tempering attack. Protection settings and information in the communication network are to be confidential and prevent any unapproved disclosure.
6. A6; the transmitted GOOSE message can be captured and modified by employing a GOOSE Forgery or Spoofing attack. A false message can lead to the failure of the substation.
7. A7; process bus communication is compromised, and useful information can be used for cyber-attacks. The configuration of the communication path is to be strictly controlled as the protection scheme of the substation is critical. Security measures are to be implemented so that vulnerabilities of the subscriber device are not exploited.
8. A8; sampled values from the merging unit can be changed by employing a Sampled Value Stream Forgery attack. Data integrity is to be implemented to prevent unauthorised data exploitation.
9. A9; compromise the firewall and gain access to the power system.

Based on these potential cyber-attacks, the substation environment must be secured against any security threats. The transmission of false GOOSE messages can cause IED settings to be modified and hence disrupt field device operations. As such, the operators of IEDs in the substation need identification and authentication; access must be controlled whether locally or centrally.

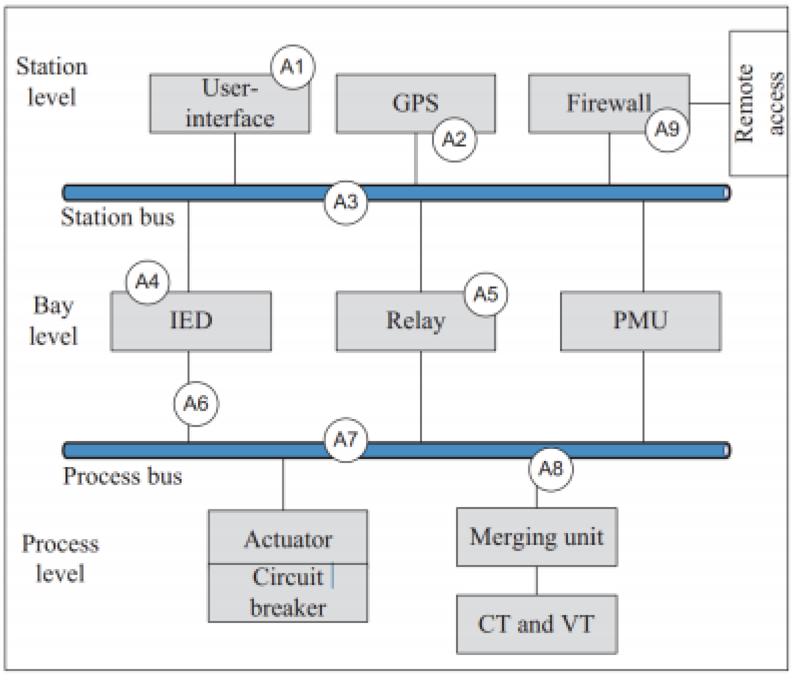


Figure 2.12: Potential Cyber-attacks in a SAS (Hong et al., 2014)

2.5.3 Synthesis and Analysis of Literature

The Internet has radically transformed the way we communicate with each other and share information. While security is a key component of IoT implementation, it is sometimes overlooked during system development. Designing with security in mind can save time and effort, as well as financial implications that could cause damage.

The overarching theme is that considering security at the design phase can save not just time and work later on, but also potential embarrassment and financial loss. Security education is a rapidly growing field in and of itself. This domain recognises that even the most secure technologies are only as secure as their users. User education educates individuals on how to defend themselves against cyber hazards.

Emerging technologies pose new challenges and opportunities for cybersecurity to address and evolve. Changes in how businesses keep data digitally, the volume of content published via the web, and the increasing number of linked devices all result in new forms of vulnerabilities. The amount of data breaches has risen. New technology entails new attack channels for cyber dangers, as well as implications for physical security. As more industries migrate to the internet and become a part of the digital world, cybersecurity is becoming a bigger field. This means that practically every industry requires cybersecurity, in addition to cybersecurity being an industry in and of itself. Cybersecurity as a field involves a diverse set of abilities that function in concert. Data and communications must be secured to ensure the security of essential infrastructure and a stable electricity supply. The IEC 62351 standard provides a mechanism for achieving that security that is widely recognised. And compliance with it ensures performance and interoperability, which increases the appeal of systems and components to network operators. As a result, a comprehensive security model is necessary for substation communication. IEC Technical Committee (TC) 57 WG 15 produced the IEC 62351 standard, to resolve substation communication security challenges. To implement cybersecurity measures in IEC 61850-compliant smart grids, a detailed understanding of the IEC 62351 standard is required. Following that, customised solutions for ensuring secure communication in various aspects of smart grid operation can be devised (Hussain, Ustun, et al., 2020).

To ensure the safety of different substation communication services, such as IEC 61850, there are security requirements included in the IEC 62351 standard. IEC 61850 is becoming one of the extensively utilised standards for substation automation. As a result, (Hussain, Ustun, et al., 2020) present a detailed examination of security concerns, cyber threats, and recommendations for IEC 61850 communication. Additionally, extensive study is given to the security issues specified in IEC 62351 for the protection of several IEC 61850 communications (GOOSE, SV, R(Routable)-GOOSE, R(Routable)-SV, and MMS). To better understand these cybersecurity techniques and put them into perspective within the context of substation automation, a brief overview is provided. Additional information is provided by the authors, including the review of the cybersecurity techniques defined in the IEC 62351-6:2007 standard with the IEC 62351-6:2020 standard.

(Yoo & Shon, 2016) makes it clear that the IEC 61850 standard is at the centre of the industrial control environment. As such, the connections between these heterogeneous protocols are inevitable. However, there may be an infringement of end-to-end security at connection points between communication protocols. As such, vulnerabilities are exposed if we rely solely on existing security requirements for individual protocols. (Yoo & Shon, 2016) studied safety issues that may arise in the smart grid and provided advice on security measures that can be taken.

(Harbi et al., 2019) further investigates authentication and key management mechanisms for securing the transmission of data on the Internet of Things (IoT). The fundamentals of IoT are agreed on regarding the change it brings into society by allowing various equipment to connect and access an internet cloud, hence attracting worldwide attention. The challenges of ensuring security and privacy in network communication in IoT is a fundamental problem. (Harbi et al., 2019) propose a secure Wireless Sensor Network (WSN) authentication and key management scheme to secure data transmission. The scheme examined security flaws such as replay attacks, denial of service (DoS) attacks, impersonation attacks, lack of mutual authentication, and session key agreement. Once the security flaws are identified, (Harbi et al., 2019) propose a secure and enhanced scheme to overcome such challenge weaknesses. Various logic and tools such as the Burrow-Abadi-Needham logic, Automated Validation of Internet Security Protocols, and Application tools are utilised for verification of the enhanced security scheme. (Karnati, 2020) further studies and discusses the cyber-attacks initiated on SV as follows:

1. Replay attack: This is a cyber security attack where cybercriminals monitor the communication network to intercept and delay (playing back) critical information such as SV packets that contain values of current and voltage.
2. Spoofing: The original SV message was attacked, captured, modified, and injected with malicious information. Figure 2.13 shows an example of the spoofing attack where the original SV message is manipulated. The subscribing device discards the original SV message stream and subscribes to the spoofed message stream. The time

synchronisation information and measurement values can be manipulated. Attackers can also tamper with the operation of the field equipment (Karnati, 2020).

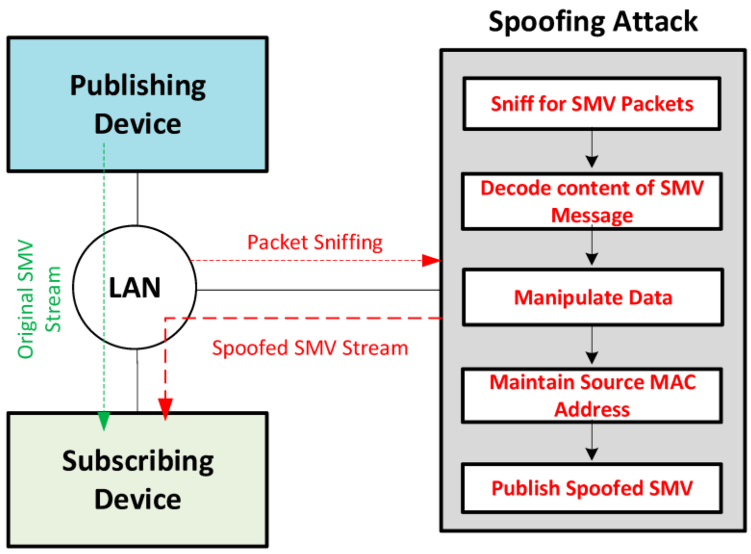


Figure 2.13: An example of spoofing attacks for SV messages (Hariri et al., 2019)

3. Flooding Attack: Attackers identify the critical information of the initial SV messages in the communication stream. The original message is then manipulated and reproduced, flooding the process bus with SV messages. The main object of this malicious attack is to attack the normal SV subscriber function where the protection functions cannot be processed.
4. High smpCnt Attack: If an SV subscribing device repeatedly gets a high number of SV messages that contain smpCnt, the device will ignore all other normal SV messages. The intruder has the potential to interfere with the regular operation of the SV in the power system. Because of this, in the long run, the usual monitoring that is done on the measurement function of the subscribing device will be discarded (Karnati, 2020).

IoT has been recognised as a revolutionary technology of the present century. To encrypt the data being transmitted, the transmission of data over a public network needs to be protected and thus a secret key should be exchanged between the communication parties. Only the authorised or involved communicating parties can access the transmitted data hence the requirement of authentication and key agreements (Harbi et al., 2019).

The convergence of information technology (IT) and operational technology (OT) environments, which were formerly designed to run separately, has evolved dramatically as digital transformation has accelerated over the last decade. This convergence, fueled by a rising reliance on cyber-physical systems and technical breakthroughs, has resulted in indisputable commercial advantages such as increased efficiency, sustainability, and creativity. However, it has also created new risks and concerns, particularly in cybersecurity.

IT environments have typically focused on managing and processing data, using techniques like encryption and firewalls to protect confidentiality, whereas OT environments prioritize managing and controlling physical devices critical to production, prioritizing integrity. As these previously independent environments merge, efficiency and visibility improve; yet, issues occur due to differing security needs and unique cyberthreats encountered by IT and OT systems.

Increased connections between IT and OT systems expand the attack surface for hackers, demanding specialized security controls and communication between IT and OT teams. IT and OT systems have significantly distinct security requirements and confront various cyberthreats, allowing IT and OT operations inside a business to become contaminated. Traditional IT security solutions may be ineffective for OT assets because they function in real-time and cannot tolerate the latency associated with IT systems. Incompatibilities in hardware, software, and communication protocols between IT and OT systems can cause disruptions with rapid and serious consequences.

In IT/OT convergence, successful cybersecurity frameworks necessitate more collaboration between IT and OT teams, as well as solutions capable of protecting all important assets inside the environment. This collaboration protects the security and stability of critical infrastructure in OT sectors, as well as the protection of sensitive information in IT systems. As a result, properly protecting convergent environments requires tackling both IT and OT's distinct security requirements and concerns.

Several authentications and key agreement schemes have been proposed in recent years to secure sensor networks within the IoT context. (Harbi et al., 2019) studied various schemes that addressed security flaws some were ineffective, and their protocols were unsafe under various attacks. In one of the papers presented by (Shen et al., 2018) where they designed two authentication and key management protocols. Both protocols are based on Elliptic Curve Cryptography (ECC) and Message Authentication Code (MAC) which generally offer confidentiality, integrity, and authenticity. However, the ECC and MAC were found by (Harbi et al., 2019) to be vulnerable to various attacks and the proposed schemes failed to achieve mutual authentication. (Harbi et al., 2019) further studies an effective WSN network architecture and a robust authentication protocol for the secure transmission of the data provided by (Amin et al., 2018). The proposed scheme by (Amin et al., 2018) provides authentication and key management. Yet cost-effectiveness is ineffective. Not limited to the review of papers, (Harbi et al., 2019) review a paper authored by (Mehmood et al., 2017). (Mehmood et al., 2017) proposed a secure mechanism called an inter-cluster multiple key distribution schemes (ICMDS) for WSNs. They focused on key management and authentication. They employed various key methods of distribution to secure communication between clusters. The proposed scheme, however, is insecure since it is vulnerable to several types of attacks.

To secure communication networks it is important to design authentication and key agreement mechanisms. As discussed, key agreement schemes for authentication and session are based on ECC and rigid one-way functions. The ECC gives greater security and is suitable for restricted environments. In the paper, (Harbi et al., 2019) discuss and review the (Mehmood et al., 2017) scheme and its security weakness issues. An enhanced ECC-based scheme is being proposed to overcome the security flaws provided by (Mehmood et al., 2017) and analyse, assess, and evaluate its security and performance.

The Mutual Authentication and Key Agreement (MAKA) is called the enhanced scheme. It consists of five phases: initialisation, key generation, node registration, authentication of nodes, and agreement to the session key. (Harbi et al., 2019) conclude informedly that the proposed authentication and key management scheme for IoT applications are more secure and effective.

(Rodriguez et al., 2021) further evaluates a fixed latency architecture to secure GOOSE and SV messages to provide authentication and confidentiality using the AES-GCM algorithm. The authors implement and test the test to successfully authenticate and encrypt real-time GOOSE and SV data in less than 7µs. The implemented design complies with IEC 61850 and IEC 62351 respectively where IEC 61850 data is generated, transmitted, and processed in less than 3ms. Data authentication and encryption can be implemented for GOOSE and SV using the AES-GCM algorithm that introduces fixed latency. Table 2.8 shows various security algorithms employed to evaluate and verify compliance with IEC 62351 standard, and time requirements to secure GOOSE. The proposed algorithm by (Rodriguez et al., 2021) is the only solution compliant with IEC 62351-6. The RSA algorithm is not compliant because of the latency time, and the combination of AES & and SHA-256 algorithms is not compliant because of the format.

Table 2.8: Comparison of IEC 62351-6 GOOSE and SV frame security implementation proposals (Rodriguez et al., 2021)

Algorithm	Functionality	Implementation	Maximum Latency (ms)	Delivery Time Usage	Maximum Throughput	Fixed Latency	IEC 62351-6 Compliant
RSA	Authentication	Software	4	133%	-	No	No (Time)
RSA	Authentication	Hardware	1.917	63.9%	-	Yes	No (Time)
RSA	Authentication	Software	6	200%	-	No	No (Time)
RSASSA-P	Authentication	Software	0.942	31.4%	-	No	No (Time)
KCS1-v1_5	Authentication	Software	3.56	118.7%	-	No	No (Time)
EtM (AES & SHA-256)	Authentication and Encryption	Software	0.242	8.07%	-	No	No (Format)
E&M (AES & SHA-256)	Authentication and Encryption	Software	0.235	7.83%	-	No	No (Format)
MtE (AES & SHA-256)	Authentication and Encryption	Software	0.284	9.47%	-	No	No (Format)
AES-GCM	Authentication and Encryption	Hardware	0.006	0.23%	> 1Gbits^{-1}	Yes	Yes

Confidentiality has proven to be difficult to implement since using asymmetric cryptography is a challenge to secure real-time traffic. Encryption is a desirable feature however presents an additional challenge of computational overheads, which need to be limited to not compromise the delivery time of the IEC 61850 data. As such (Rodriguez et al., 2021), study the current security attacks, and further evaluate available cybersecurity solutions. (Hohlbaum et al., 2010), have studied the RSA algorithm, using a software implementation and the results show that asymmetric cryptography is a challenge for securing real-time traffic. The minimum time required to generate the digital signature is 1.5ms to 4ms, which is not compliant. IEC 62351-

1 recommends that encryption algorithms not be used with IEDs due to their high processing times. (Ishchenko & Nuqui, 2018) further confirmed the results using the latest technological hardware, and the results were still not in compliance. (Farooq et al., 2019) propose an alternative algorithm for securing real-time traffic. The authors implemented the RSASSA-PSS digital signature using Python and verification times were calculated. The authors found that neither RSASSA-PSS digital signature is fast enough and unsuitable to secure critical operations for securing IEC 61850 data. However, the results show that the RSASSA-PSS digital signature has better and improved performance than RSA.

The smart grid is a rapidly evolving power system of generation systems, distribution systems, transmission lines, electrical equipment, and control technology to meet demand in the 21st century (Kim et al., 2013). The smart grid is an electrical grid comprising controls, computers, automation, and new technologies that respond digitally to changing electrical demands. Distributed communication and intelligence capabilities can improve the efficiency and reliability of the smart grid and other networks. However, if smart grids are not implemented with sufficient protection controls, they could create new vulnerabilities that would allow hacking and cyber-attacking of utilities.

Thus, (Kim et al., 2013) evaluate the safety output of the GOOSE, implementing the IEC 62351-6 MAC protocol, applying it to F-IED using the IEC 61850 GOOSE and Hardware Security Module (HSM), and develop the environment for the security test beds. As the Electric Power Research Institute (EPRI) reports, one of the major issues confronting the development of smart grids is the cyber security of networks. According to the EPRI Report, *“Cyber security is a critical issue due to the increasing potential of cyber-attacks and incidents against this critical sector as it becomes more and more interconnected. Cyber security must address not only deliberate attacks, such as from disgruntled employees, industrial espionage, and terrorists, but inadvertent compromises of the information infrastructure due to user errors, equipment failures, and natural disasters. Vulnerabilities might allow an attacker to penetrate a network, gain access to control software, and alter load conditions to destabilise the grid in unpredictable ways.”* (National Institute of Standards and Technology, 2012)

IEC 61850 and IEC 62351 are supported by (Kim et al., 2013). However, it must be noted that IEC 62351 is not only for IEC 61850. It includes cybersecurity for IEC 61850, IEC 60870-5 (101/104 and DNP3), IEC 60870-6 Inter Control Center Protocol (ICCP), and IEC 61968/61970 Common Information Model (CIM). The design and configuration specification for substation automation is IEC 61850, as shown in Figure 2.14.

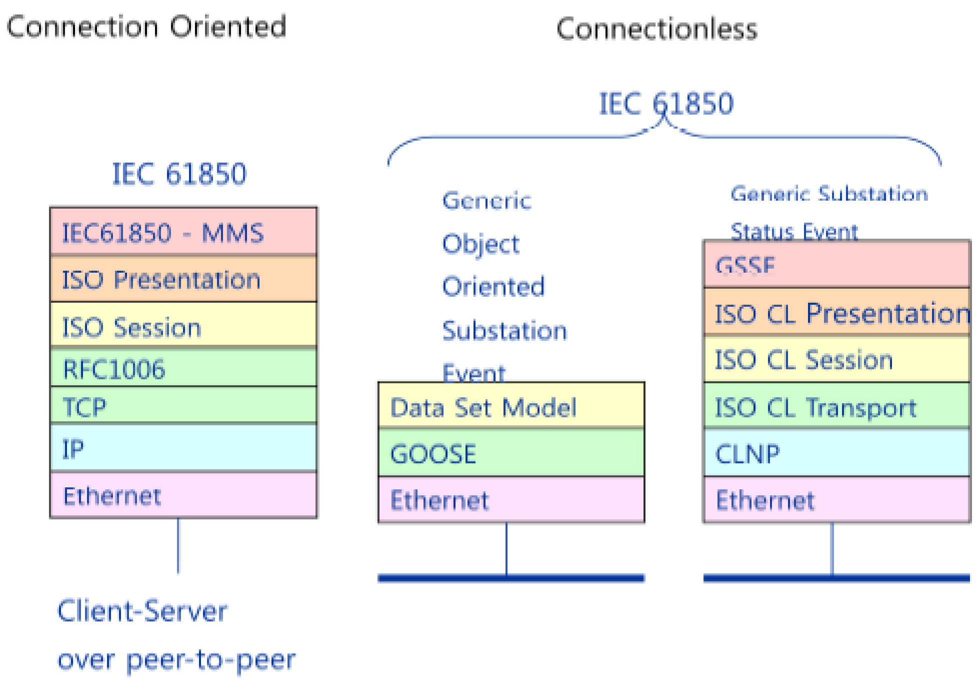


Figure 2.14: IEC 61850 Communication Model (Kim et al., 2013)

IEC 61850 utilised object-oriented data models to explain the details of various equipment and automation functions. The IEC 61850 specifies the interface between the IEDs and the schemes that map them to multiple protocols using TCP / IP and high-speed ethernet. GOOSE is a protocol intended for use in IEC 61850 for sending critical time messages, such as substation incidents, commands, and alarms inside the substation network as mentioned by (Kim et al., 2013). Because GOOSE is designed to transmit time-critical messages; messages must be sent within 4ms, so security mechanisms that affect transmission rates are inappropriate. IEC 62351-6 provides security measures for authentication and integrity measures which include critical timing requirements for digitally signing the messages.

(Kim et al., 2013), used MAC mechanism IEC 62351-6 as illustrated in Figure 2.15. The MAC mechanism includes hash-value calculations, authentication-value calculations, authentication-value decryption, and digital signature verification from sender to receiver.

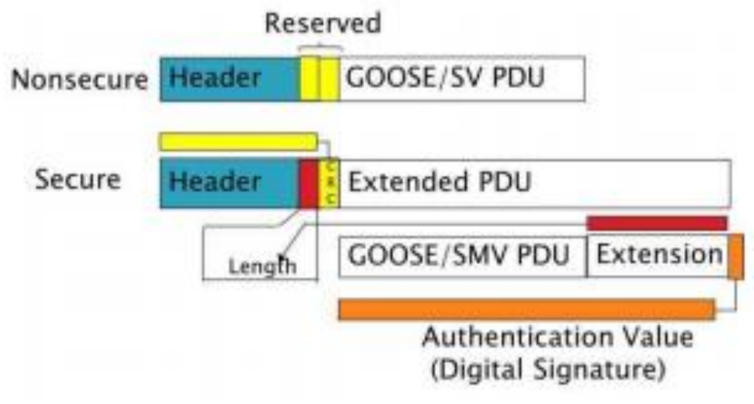


Figure 2.15: Secure GOOSE Procedure (Hussain et al., 2019)

(Hussain et al., 2019) agree that IEC 61850 is a very popular standard that is currently being researched intensively and note that it is also relevant to study aspects of cybersecurity. (Hussain et al., 2019) develop the S-GoSV (Secure GOOSE and SV) software which generates GOOSE and SV messages. The software incorporates a security feature to protect them from cyber-attacks inside a substation. The Secure GOOSE and SV software implement RSA digital signature algorithms as IEC 62351-6 stipulates. The generation and verification procedure used for the RSA digital signature algorithm is shown in Figure 2.16. The full process of generating and verifying digital signatures is described in (Farooq et al., 2019). However, RSA digital signature algorithms take a long time based on performance studies and do not comply with critical time requirements as set out in the substation communication standard IEC 61850. To address this concern (Hussain et al., 2019), a Keyed Hash-Message Authentication Code - Secure Hash Algorithm (HMAC-SHA256) was developed and implemented in the S-GoSV message securing software. Furthermore, the authors discuss GOOSE and SV message structures as stipulated by IEC 61850-8-1 and IEC 61850-9-2, and demonstration results with Wireshark are provided.

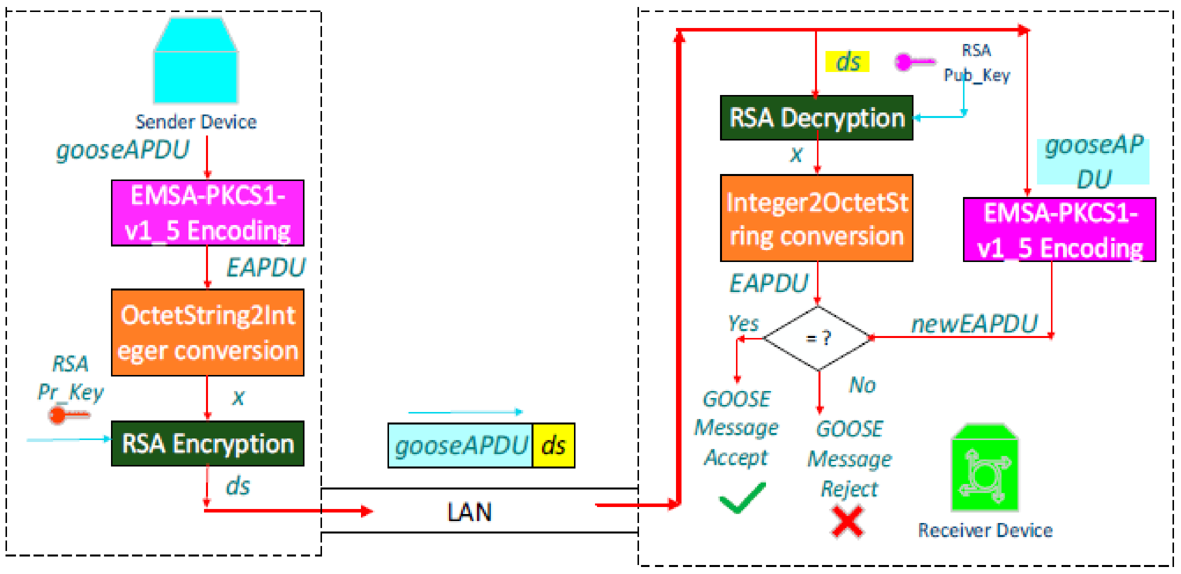


Figure 2.16: Generation and Verification of RSA Digital Signature Algorithm (Hussain et al., 2019)

(Wang et al., 2019) presents a key management security mechanism for substation automation. The authors clearly outline that the security of encrypted messages, cryptographic systems, and protection of the keys are a unit and depend on each other. Key management schemes are based on key generation, key negotiation, key distribution and key updating. (Wang et al., 2019) implement an invulnerable key management algorithm to ensure the protection of keys for cryptographic systems and messages. They propose an improved key management mechanism which has a great advantage in terms of communication. The proposed key generation algorithm meets the confidentiality requirements of message security, the stability of the communication network is better, and the real-time performance is better. Security analysis was presented and different attacks like the MITM attack, replay attack, and anti-tampering have been experimented with, and each attack has been prevented.

(Kriger et al., 2013) examined the GOOSE message in a substation that conforms to the IEC 61850 standard and confirmed their results using simulation and a practical experiment was conducted with IEDs. The IEDs were utilised to generate GOOSE messages and Wireshark was used for analysis. The authors concluded their results with an actual testbed confirming the GOOSE PDU as specified in Part 8-1 of the IEC 61850 standard.

2.6 Overview of Raw Socket Programming

As previously mentioned above, a socket provides an abstraction layer for the programmer to send and receive data, either using two processes on the same machine or across the network to a different device. We can then define socket programming as a method of connecting two network nodes so that they can communicate with one another. One socket listens on a specific port at an IP address, and another socket reaches out to the other to establish a connection. While the client attempts to contact the server, the server creates a listener socket. (Socket Programming in C/C++ - Geeks for Geeks, 2019)

In this thesis, socket programming will be conducted in C using TCP/IP. A TCP refers to a connection-oriented communication protocol. In socket programming, a connection between two processes is referred to as an association. As such, the association can be defined as a data structure that specifies the two processes and their method of communication. The data structure includes the protocol, addresses, and processes.

In the OSI model, the protocol is a layer that is included between the application and the internet protocol layer. TCP is a protocol utilised in networking for sending data packets. It guarantees that data reaches the intended recipient. Before delivering the data packets, a connection is thereby established at the source and destination nodes. The connection is then maintained until the transmission of data is complete. Subsequently, TCP has a retransmission feature, which means that when a TCP client delivers data to the server, it expects an acknowledgement in return. If an acknowledgement is not received, the data will be lost after a given length of time, and TCP will automatically retransmit the data. TCP is a dependable stream because of the connection notion, which allows faults to be identified and compensated for by resending failed packets.

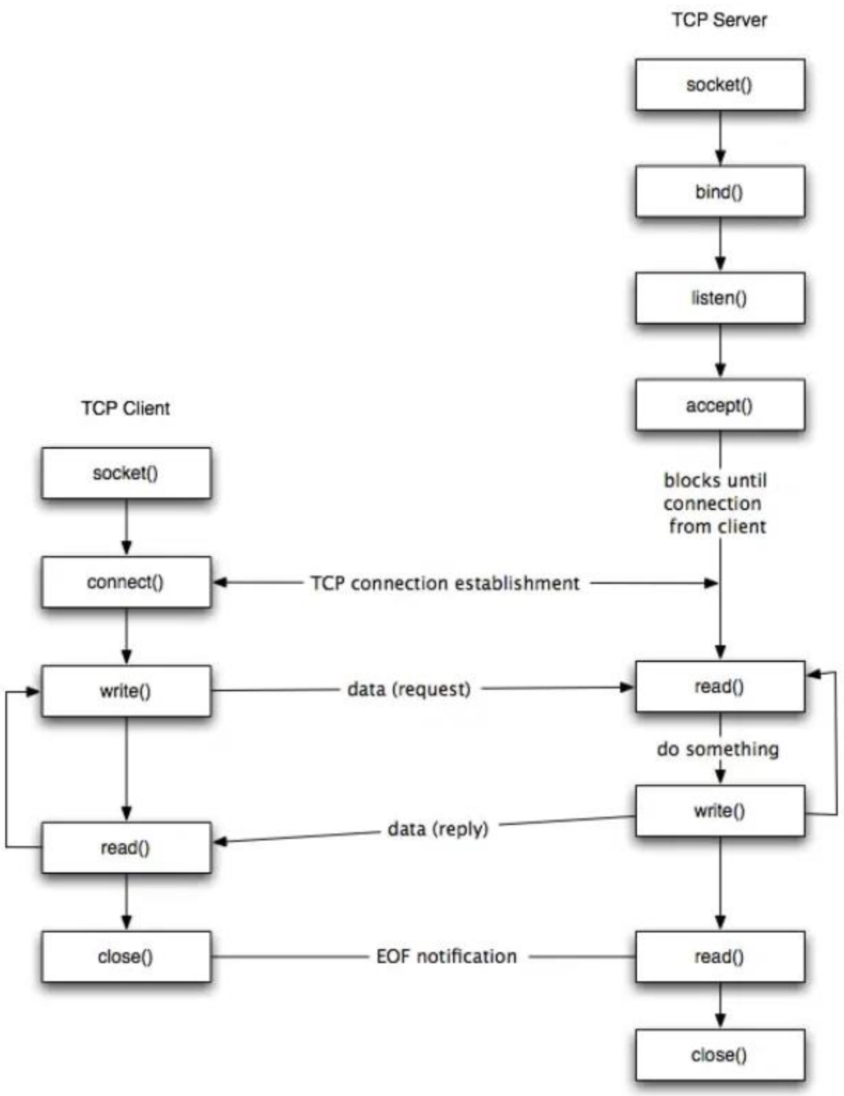


Figure 2.17: Sequence of function for client-server communication (Socket programming in c using TCP/IP - Article world, 2021)

In the TCP/IP model, network communication incorporates a client-server topology. Specifically, communication is initiated, and a connection is established between the client and the server. Examples of client-server communication will be implemented under Section 4.2. The implementation will specifically follow a sequence of functions for client-server communication as shown in Figure 2.17. Subsequently, after establishing a connection with a client, the server will wait for the client to send a message. After receiving the message, the server will examine it and send an appropriate response in accordance with the message.

Steps to creating client communication using TCP are as follows (Socket programming in c using TCP/IP – Article world, 2021):

1. *“Create a socket using the socket () function in c,*
2. *Initialise the socket address structure as per the server and connect the socket to the address of the server using the connect (),*
3. *Receive and send the data using the recv () and send () functions,*
4. *Close the connection by calling the close () function.”*

Steps to creating server communication using TCP are as follows (Socket programming in c using TCP/IP – Article world, 2021):

1. *“Create a socket using the socket () function in c,*
2. *Initialise the socket address structure and bind the socket to an address using the bind () function,*
3. *Listen for connections with the listen () function,*
4. *Accept a connection with the accept () function system call. This call typically blocks until a client connects to the server,*
5. *Receive and send data by using the recv () and send () functions in c,*
6. *Close the connection by using the close () function.”*

The client-server mode is an information-sharing mode that is commonly used in information systems, such as databases. The most fundamental aspect of client-server setup is the custom and the server. The term "client" usually refers to a personal computer or a workstation. It presents the terminal client with a very user-friendly interface, such as Microsoft Windows and other similar programs. The server offers the client a group of users who are all using the same service application, which is provided by the server. The database server is the most widely used type of server. It allows a large number of clients to share the same access to information sources. (Xue & Zhu, 2009)

Figure 2.18 shows the client-server system structure. Computer systems with a client-server structure can exchange information and resources across them. For example, files and disk space can be shared between systems, while processors can collaborate and transmit messages inexorably amongst numerous processors. Some of the processors are running on the client side, while the rest are working on the server side of things. The distribution of task-level applications between client and server lies at the heart of the client-server system's structure. The software that facilitates communication serves as the foundation of the exchange. The TCP/IP protocol suite is an example of this type of software. As a result of supporting this software (communication software and operating system), the primary objective is to create a fundamental structure for distributed applications to operate on. (Xue & Zhu, 2009)

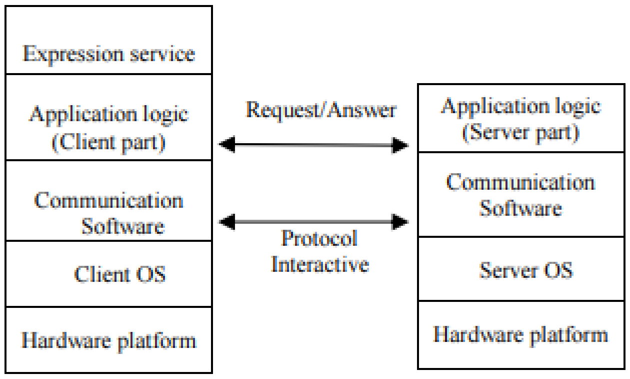


Figure 2.18: Client-server system structure

When we use TCP protocol to connect a client and a server, we gain several advantages. For example, TCP is well-suited for applications requiring high reliability, and transmission time is significantly less critical. Other protocols, such as HTTP, HTTPS, FTP, SMTP, and Telnet, make use of it. TCP rearranges data packets in the requested order. There is a 100% assurance that the data transfer is complete and arrives in the sequence in which it was sent. TCP employs Flow Control and necessitates the transmission of the handshake procedure to establish a connection before sending data.

Henceforth, The IEC 61850 standard specifies communication interfaces that enable publisher-subscriber and client-server services to be provided. These services help IEC 61850 devices communicate via MMS for client/server applications. GOOSE messaging is used to provide peer-to-peer communication between publishing and subscribing devices, while SV messaging is used to distribute measurements. Therefore, IEDs connect with the HMI via the MMS protocol's messaging capabilities. Inter-IED communication is accomplished via the GOOSE publisher-subscriber messaging system. Through a publisher-subscriber arrangement, the process-level devices communicate power system information (voltage and current measurements, circuit breaker status, alarm notifications etc.) from switchyard source devices to the bay level.

For decades, communication methods have been utilised to help power systems work better. By delivering innovative, simple-to-use, durable technology for power system protection, automation, control, and monitoring, power suppliers aim to increase productivity and make electric power more reliable, safe, and cost-effective. The strategy's major objective is to develop acceptable communication technologies and protocols (Ozansoy et al., 2007). It has become practical and justifiable to incorporate station IEDs on a peer-to-peer communication network as a result of the advent of IEC 61850, which is a standardisation effort.

The standard also allows for peer-to-peer communication for exchanging SV and GOOSE messages between devices, furthermore, the client-server topology is provided by mapping to the MMS stack. Figure 2.19 presents the peer-to-peer topology. For IEC 61850, the critical point to remember about publisher-subscriber communication is that it can occur only if a client-server architecture has been established previously. When a server is destroyed, any aspects that are reliant on it are also destroyed.

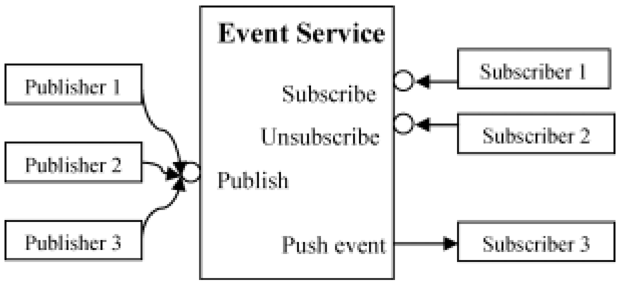


Figure 2.19 Publisher-subscriber communication model (Ozansoy et al., 2007)

IEC 61850 requires the deployment of a client-server and peer-to-peer communication architecture to enable the ACSI, one of the critical techniques for ensuring compatibility between devices manufactured by various manufacturers. Table 2.9 shows reviewed papers for socket programming. Not many papers were reviewed as the topic is not a focal study.

Table 2.9: Socket programming reviewed literature papers.

Paper	Research Objectives	Method	Outcomes
(Kalita, 2012)	The purpose of this paper is to introduce sockets and their use in network programming. In addition, the paper discusses socket programming in Java over TCP.	In client-server applications, sockets are essential. Writing to or reading from these sockets allows the client and server to communicate with one another. This paper teaches network programming concepts and elements involved in constructing network applications utilising	This paper studies sockets, ports, and socket programming over TCP, and UDP. Sockets are used in network programming to provide interprocess communication between hosts, with sockets serving as the communication's endpoint. Because the Internet protocol is used

		sockets. Java was designed to establish client-server traffic via sockets, performing socket functions is one of the most fundamental network programming tasks that a Java programmer is likely to confront.	to communicate between computers, sockets are also known as network sockets or Internet sockets. Java has outperformed all other languages in terms of establishing connections between clients and servers.
(Kumar, 2019)	The author provides a detailed study of the client-server topology, emphasising core aspects of development, implementation, and research issues. The purpose of this study is to raise awareness of client-server topology development issues and to highlight important principles.	One of the primary goals of this research is to identify research difficulties in the client-server system. Studying the performance evaluation, and the reliability of the client-server system. The goal of performance evaluation is to give a fair and comprehensive conceptual understanding of the client-server system.	The client-server system and its many components were explained in this study: client-server architecture, physical and logical components of client-server architecture, and implementation strategy. We also included some client-server scenarios. Performance, reliability, trusted system design, and secure system development are some of the developing disciplines for study and development in the current environment.
(Xue & Zhu, 2009)	The use of the client/server mode, as well as the notion and programming approach of sockets based on client-server, are discussed in this work.	The approach of software design for communication between client and server processes utilising the socket mechanism is primarily examined, and examples of connection-oriented service programs are shown. The transmission layer can provide TCP or UDP.	The client-server mode is a common information transfer paradigm in information systems. The distribution of task-level applications between client and server lies at the heart of the client/server system structure. To make it easier to put together an effective client service program. On the transport layer,

			the authors stated that they require a thorough understanding of the TCP and UDP protocols.
(Bertocco et al., 1998)	The authors present real-time communication with client-server architecture using the Secure Shell (SSH) protocol. As such, we frequently assume that Internet communication is just as reliable as traditional kinds. We anticipate that what we are saying will reach the intended receiver in its original form. Unfortunately, there are numerous security issues in distributed control systems, as well as numerous methods for the problem of transmission security.	The focus of this study is on a low-cost solution: TCP communication tunnelling using the well-known SSH protocol. The Network Control System experimental framework employed in this study entails real-time control of many plants linked to the controllers via the ZigBee network. ZigBee is a wireless technology that was developed as an open global standard to address the unique requirements of low-cost, low-power wireless sensor networks. The ZigBee technology was designed to transfer data over radio frequency in difficult environments. ZigBee allows for a large range of wireless network configurations using low-cost, low-power solutions. It increases the ability to run for several years on low-cost batteries for a variety of testing applications.	In contrast to private enterprise solutions, the SSH client technique necessitates essentially no user credentials on the client system. As a result, we demonstrate that the SSH protocol is more efficient than the TCP protocol. SSH is a network communication protocol that is designed to be controllable and cost-effective to adopt. Client and server SSH software are available for practically all operating systems. The client in this article is a computer, and the server is a Raspberry Pi board. The SSH client and server must establish a stable connection before any communication can take place. This allows them to share keys, passwords, and lastly, any data they send to one another providing strong authentication and firm communication between client and server.
(Piantadosi et al., 2015)	It has become more necessary in recent years in many scientific disciplines to collect and analyse large amounts	In the framework of an image-based medical diagnostic environment, the authors created a case study. This	The authors have designed a medical case study to aid in the discovery of lesion detection automatically.

	of data. As a result, there is a rise in the study of automated data management and data mining to develop trustworthy ways for data analysis and interpretation. The authors propose a client-server architecture for advanced distant data processing that is secure, smart, versatile, and capable of incorporating pre-existing local applications.	enabled us to address several additional issues that were also closely associated with medical data and the analysis of that data. The purpose of this research is to propose an architecture for advanced remote workflow execution in a secure and versatile client-server intelligent environment that can interact with a wide range of software applications and systems.	This design provides the radiologist with access to secure and robust draughting software. Tests have shown that the suggested design has several advantages, including an improvement in system throughput, ease of upgradeability, maintainability, and scalability, among others. The authors conclude that the work is useful for facilitating the integration of intelligent devices into existing infrastructure.
(Ozansoy et al., 2007)	Using the IEC 61850 protocol as an example, the authors offer a real-time publisher/subscriber communication model as a means of addressing the protocol's unique behaviour and communication requirements. The authors provide a full description of the model's architecture and implementation, as well as some noteworthy performance results, in their paper.	At the heart of this strategy is the development of appropriate communication technologies and protocols. The authors began their research on IEC 61850 by transforming it into a real protocol by implementing its services and information models as concrete programs using sophisticated object-oriented programming techniques. Additionally, the design and implementation of a standard delivery service were conducted, which was created specifically to address the IEC 61850 protocol's	The paper used simulations to assess the performance of the publisher/subscriber communication model. The simulations revealed that the designed architecture is capable of providing the IEC 61850 protocol with essential communication services. It adds a negligible amount of overhead to the underlying protocol stack, while yet keeping the overall delay within acceptable limits.

		communication requirements, to support client/server and publisher/subscriber communication models.	

2.7 Conclusion

In closing, to achieve interoperability, authentication, integrity, and confidentiality within the substation automation environment, IEC 61850 and IEC 62351 should be conformed to. With the above literature, it is found that the challenge of encryption cannot be successfully solved to meet timing requirements for secure GOOSE. A new authentication scheme must be developed to meet compliance with cyber security and timing requirements. In 2020, IEC issued a new version of the security standard, IEC 62351:2020. The standard introduces Secure Hash Algorithm-256 (SHA-256) and Advanced Encryption Standard (AES) Galois Message Authentication Code (AES-GMAC) algorithms. The implementation of these two algorithms is expected to meet compliance for cyber security, and timing requirements to secure GOOSE and R-SV messages. The standard also includes AES Galois/Counter Mode (AES-GCM), to allow to provide both confidentiality and integrity. (Hussain, Farooq, et al., 2020) studies and implements a method of achieving both authentication and encryption by employing Authenticated Encryption with Associated Data (AEAD). The authors study three different methods for the AEAD algorithm to achieve encryption and authenticated GOOSE messages. These are Encrypt-then-MAC (EtM), Encrypt-and-MAC (E&M) and MAC-then-Encrypt (MtE). It secures GOOSE message communication by employing AES-128 encryption and HMAC SHA-256 message integrity. (Hussain, Farooq, et al., 2020) conclude that the AEAD algorithm can be employed successfully to secure GOOSE, meeting the time requirements of 3ms. Table 2.10 shows reviewed papers for IEC 62351.

Table 2.10: IEC 62351 reviewed literature papers

Paper	Research Objectives	Method	Outcomes
(Yoo & Shon, 2016)	Studies the electrical grid in Korea and further addresses security vulnerabilities, requirements, and security mechanisms in power system	Investigating the integration of the IEC 61850 protocol with other communication protocols such as DNP3, IEC 61970, and OPC UA.	Security issues that can occur in a power grid were noted. Various security countermeasures were proposed. The security technique was employed in an IEC

			61850 – DNP3 environment and verified.
(Kang et al., 2011)	The authors propose strategies of key management for data encryption in SCADA networks and further study the issues present in SCADA networks.	Evaluating the SCADA network security problems in the Korean power grid. Encryption is employed in the SCADA system for cyber security.	Symmetric encryption is applied to the SCADA network. However, the authors found that symmetric encryption is more vulnerable to cyber-attacks than asymmetric encryption. The key distribution period based on the Quality of Service (QoS) function is evaluated, and the authors conclude it is more significant for network performance and security.
	A cyber security attack can physically impact the failure of critical infrastructure. This paper studies and evaluates the most used security techniques.	A hardware cyber-physical testbed including the security mechanism is implemented to secure the substation from security vulnerabilities.	A substation is simulated, and the Anomaly Detection System (ADS) successfully captures the malicious attacks that have been detected. ADS is applied to the testbed to evaluate the effectiveness of the detection capabilities.
(Moreira et al., 2016)	A Stuxnet virus is one of the popular security attacks that were able to maliciously intrude on the SCADA and take control of its operation in Iran. Awareness was raised and the authors explored available security techniques and their applicability to	Cyber security mechanisms such as RSA, HMAC-SHA, ECDSA, and AES algorithms are studied to prevent attacks. The MACsec cryptographic technique is also presented as an alternative as it provides hop-by-hop integrity and	The paper discusses the evolution of substation communication and further studies how IEC 61850 is integrated with the power system environment. Also, cyber security for the communication standard, IEC 61850 is discussed to evaluate

	SAS. A MAC-based security mechanism is proposed.	authenticity, but not end-to-end source authenticity.	the prevention of unauthorised access to SAS. The authors recommend that a hybrid solution be developed to include hop-by-hop group authentication and integrity using symmetric cryptography and end-to-end authentication.
(Wang et al., 2019)	This paper presents a key management method for the smart substation. The security of cryptosystems depends on ensuring that the key management is secured and not compromised. the authors propose an improved symmetric polynomial-based key management method.	The proposed key management method is analysed to verify the message security and efficiency to prevent security attacks. Simulations are deployed to test the various attacks mentioned and each security attack is prevented and resisted.	The symmetric polynomial key management scheme based on symmetric encryption is implemented. An analysis of the security scheme is presented through simulation and the results verify real-time performance.
(Schlegel et al., 2017a)	This article presents a security evaluation of the IEC 62351 compliance, examining several aspects of the security standard. The authors further discuss how the security standard can improve security in SAS.	An overview of the ten parts of the IEC 62351 standard is evaluated. The standard contains security measures to ensure that integrity, authenticity, and confidentiality are implemented in a substation environment.	The standard addresses security issues present in SAS. Also, discuss the performance related to backward compatibility. The authors are satisfied with the standard as it provides a reasonable amount of security. However, some of the cryptographic algorithms also need to be considered.
(Hohlbaum et al., 2010)	In substation automation, IEC 61850 standard and IEC 62351 cyber security standard	The performance requirement of the IEC 62351-6 standard has been confirmed to not	The authors further look at the available methods to secure the SAS. Since IEC 61850 is

	need to be implemented. IEC 62351 is the security standard for IEC 61850 communication. In this paper, the challenge of addressing secure communication in the power system in compliance with the communication standard is presented. IEC 62351-6 of the standard is discussed (in particular) to address the performance requirements for IEC 61850 protocols.	be compliant with IEC 61850 GOOSE and SV data delivery.	presented which makes use of ethernet and TCP/IP, firewalls and VPN technology can be used to protect the security of remote stations and systems. Systems and devices must further use user authentication and authorisation.
(Hoyos et al., 2012)	The authors demonstrate a practical experiment by attacking the GOOSE protocol on cyberinfrastructure.	Malware is created to capture and tamper with the GOOSE message. The GOOSE message is then reinjected into the power grid to exploit the network.	Mitigation of the malware attack was discussed. Some mitigation measures include the introduction of strict IT techniques to be put in place. In the power grid, devices require security algorithms to be implemented. Data needs to be encrypted and authenticated. Utility infrastructure must not only be secured physically but also cyber security techniques are to be deployed.
(Farooq et al., 2018)	The communication requirements for the phasor measurement unit (PMU) are stated in the IEEE C37.118.2005 and the IEC/TR 61850-90-5:2012. The standard establishes data	Due to the number of attacks on critical infrastructure, the implementation of a security technique is a critical requirement in PMU networks. The IEC 61850-90-5 addresses	It's paramount that the communication network be secured from any security attacks. The implementation of authentication is critical to protect devices in the communication network.

	communication between PMUs etc. The transport data needs to be compliant with IEC 61850. To secure PMU communication networks from MITM attacks, the paper presents a certificate-based authentication system.	the security issue and proposes an HMAC algorithm together with a key distribution centre (KDC) for authenticating and providing encryption to data. However, the standard does not address the impact of the man-in-the-middle (MITM) attack during KDC.	The authors developed an authentication method to mitigate MITM attacks during KDC key exchange in PMU networks. However, the authors have noted that IEEE C37.118 does not address cyber security concerns and IEC 61850-90-2 only addresses certain security techniques, without detail of implementation.
(Tebekaemi, 2016)	The authors develop and implement an IEC 61850 compliant substation testbed simulation for cyber-physical security studies. As previously noted, current the substation environment has evolved in a revolution of intelligence introducing information and communication technology. As such, to maintain a secure operation of the substation, the various security concerns must be assessed to develop a robust security solution.	In the paper, the physical vulnerabilities and cyber vulnerabilities are discussed. The power system is modelled in Matlab/Simulink. The simulation includes the power model, IED model, communication model, and attack model. The IED must be IEC 61850 compliant. The process level includes the switchgear devices, bay level, and station level. The switchgear devices communicate with the bay-level IEDs via GOOSE messages with HMAC for security. The communication model consists of the process network and the station network that supports GOOSE and SV protocols. The attack	A simulation testbed is being used to perform analysis of the power grid and security protocols. It further provides an overview of physical attacks on the grid. The physical system and communication network must support the IEC 61850 standard. The simulation testbed is tested against network attacks at the process and network LAN. Using Wireshark, the authors were able to capture and manipulate the GOOSE messages. The MITM attack was also deployed to intercept messages. The authors clarified that the simulation was tested without compliance with the IEC 62351 standard. As such, the simulation

		model can be a physical attack on the system or a network attack.	was then performed introducing IEC 62351 and the attacks were not successful.
(Hong et al., 2014)	The proposed NIDS uses multicast messages to monitor and identify suspicious activities in substation automation systems. It uses a specification-based algorithm to detect and prevent cyber-attacks.	The performance test was performed for various scenarios related to cyber intrusions. The objective of the test was to evaluate the reliability and correctness of the proposed intrusion detection system. The cyber-attacks were simulated on a testbed and the results show that the proposed algorithms can identify anomalies and prevent attacks in substations.	The experiment was carried out for various security attack scenarios, such as packet modification, denial-of-service attacks, and replay attacks. This paper presents an intrusion detection system that can identify anomalous behaviours in real-time environments. It can also be used to prevent the exploitation of the network by detecting malicious activities carried out by multicast messages.
(Strobel et al., 2016)	The authors present the weakness in the IEC 62351 standard. These vulnerabilities allow for the replay of GOOSE and SV messages. The other weakness present in the Simple Network Time Protocol (SNTP) protocol also makes the system vulnerable to security attacks.	This paper presents the findings and recommendations related to the issues with IEC 62351. The authors analyse part specifications of the IEC 61850 and IEC 62351 standard.	The authors analysed and discovered the weaknesses. The weakness identified is the replay attack in the GOOSE message, cross-receiver replays in the SV message, and the attacks on the SNTP. They reiterate IEC 62351 needs to be improved as IEC 61850-based infrastructure demands a robust security technique.
(Wright & Wolthusen, 2016)	This paper discusses IEC 62351-3 standards for public-key management.	An analysis is provided to clarify that IEC 62351-3 does not align with the performance requirements in IEC 61850. The authors	IEC 62351-3 implementation of TLS and public key management is discussed. As stated, for the TLS algorithm to

		effectively make it clear that this may undermine operations. As such, operations may be vulnerable to denial-of-service attacks. The data packets transported using the TLS protocol are secured as per IEC62351-3, but this is contingent on the effective deployment of public key infrastructure.	function, there must be a public key algorithm to allow authentication. Possible problems are stated regarding the implementation of the public key infrastructure. In the same manner, possible solutions to these problems are addressed.
(Kim & Kim, 2014)	An evaluation of the IEC 61850 standard is conducted. Secure communication to the protocol is implemented using IEC 62351-6 and IEC 62351-4.	An implementation of the IEC 62351-6 and IEC 62351- security mechanism is employed on a Smart Distribution Management System (SDMS) that uses IEC 61850 communication.	To secure GOOSE, the MAC-SHA256 security algorithm and ECDSA digital signature are implemented. To secure MMS, IEC 61850 MMS stack, ECDH, ECDSA, AES 256, and SHA algorithms are used.
(Hong et al., n.d.)	Cyber security has become an issue in the newest technology used in SAS. The modernised network uses enhanced communication functionalities. The authors design a security algorithm to provide authentication and integrity for securing substation operations. Malicious GOOSE and SV communication can cause the power system to fail.	A security technique is developed to ensure authentication and integrity. The security algorithm is to meet the performance requirements imposed by IEC 61850 and IEC 62351. The test is conducted on a computer that includes an embedded system. The objective is to prevent spoofing and replay attacks and ensure authorised access.	MAC is utilised to confirm the authenticity and integrity of a message. As an added feature, key distribution is employed in peer-peer communication. However, adding an encryption algorithm increases the message processing time. Therefore, encryption is to be left out unless confidentiality is required in the message. The authors concluded that the HMAC technique outperforms alternative authentication mechanisms.

According to the analysis of the literature, several research projects are examining the use of GOOSE and SV messages in substation automation systems. Additionally, data security approaches are examined to ensure the integrity of IEC 61850 protocols. Security requirements are identified and factors such as data integrity, authentication, confidentiality, performance, and compatibility with the IEC 61850 standard are considered. Factors such as the algorithm's strength and vulnerabilities are assessed, including performance implications. As a result, this thesis will contribute to the effective transmission of knowledge about the IEC 61850 standard, in particular the R-SV and GOOSE messages, and the IEC 62351 standard. I researched the various options and determined that Authenticated Encryption is a secure algorithm. Authenticated Encryption integrates data encryption and authentication into a single algorithm, giving a comprehensive solution for protecting data integrity and authenticity. Chapter Three of this thesis delves further into the IEC 61850 standard, focusing on the information mapping from communication services to GOOSE and SV messages. It further discusses in detail the GOOSE and SV device models and their message structure.

3. CHAPTER THREE: THEORETICAL FRAMEWORK

3.1 Introduction

The IEC 61850 standard is the result of many industry initiatives to standardise communication in substation automation. Since the early 1990s, non-standard communication mechanisms have been utilised in SCADA systems, although they lack interoperability and interchangeability. This was because multiple remote terminal units (RTUs) from different vendors didn't support the same protocol. With such communication systems, RTUs were single-purpose devices, making distributed functions difficult. To create a substation automation system based on the IEC 61850 standard, a thorough examination of the standard is required. The IEC 61850 standard was created to describe application objects that can be sent via data communications. Years of development in automation and protection object modelling followed. In addition to the literature discussed in Chapter 2, we can effectively view that the IEC 61850 standard was introduced to set out the standardisation of communication in substation automation systems (SAS), including smart grids. Accordingly, the development of the IEC 61850 standard has been identified based on various advantages over hardwired and legacy communication techniques. The advantages include the virtualisation of the substation automation that is flexible, interoperability between devices, and reduction of engineering time (Abdolkhalig, 2014). In addition to providing a strong network of architectures, the IEC 61850 standard sets rigorous testing criteria for substation equipment such as fast communication and common communication protocol between field devices.

To develop an authentication algorithm for Generic Object-Oriented Substation Event (GOOSE) and Sampled Value (SV) messages, in-depth knowledge of the IEC 61850 standard is required. This chapter introduces the IEC 61850 standard for GOOSE and SV communications and provides an overview of the IEC 62351 standard. This chapter is divided into three parts: Section 3.2 provides an overview of the IEC 61850 standards that are relevant to this study. In Section 3.3, the IEC 61850 process bus and mapping data to SV and GOOSE messages are studied. In Section 3.4 and Section 3.5, we discuss the security challenges and requirements for IEC 61850, Section 3.6 provides an overview of the IEC 62351 security standard.

3.2 Overview of the IEC 61850 standard

The IEC 61850 standard consists of ten sections where the large sections are further split into smaller sections resulting in the entire standard comprising a total of fourteen parts. Necessary for this research, Part 5, Part 6, Part 7, Part 8, and Part 9 will be studied. The IEC 61850 standard describes the requirements for substation automation and establishes a future-proof architecture for substation automation that enables interoperability, scalability, flexibility, and maintainability of substation automation systems.

3.2.1 Part 5 – Communication Requirements

The IEC 61850 standard provides performance classes for the various types of messages required to map data (data objects and attributes) to specified protocols in Part 5. This standard defines five communication profiles, three of which are for time-critical communications known as link-layer communication services, while the other two are for non-time-critical messages. These communication profiles are shown in Figure 3.1.

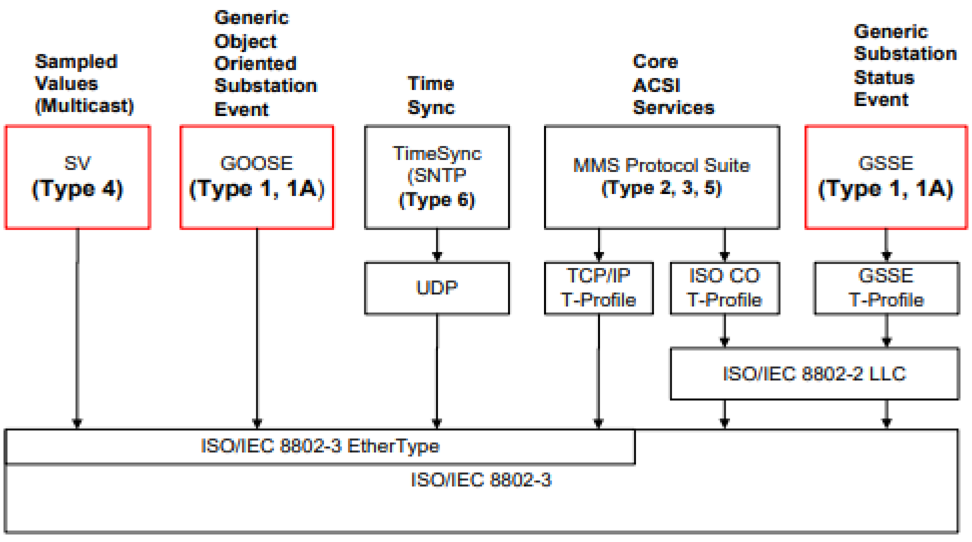


Figure 3.1: IEC 61850 message types and performance class (Ncube, 2012).

Figure 3.1 depicts three time-critical communication services, each of which is indicated in red blocks. These are the GOOSE, Generic Substation State Event (GSSE), and SV messages, among others. Since these communication services are mapped directly to the data-link layer, protocol overhead is reduced, and performance is increased. The two remaining communication profiles described in this standard are the device time synchronisation messages based on the simple network time protocol (SNTP) protocol and the Manufacturing

Message Specification (MMS) profile for the management of substation equipment (Ncube, 2012).

The Type 1, Type 1A, and Type 4 time-critical messages, each of which is mapped to a distinct EtherType, are of special interest to the researchers working on this project. Deliveries times for different types of messages are established in Part 5 of the standard; thus, all Intelligent Electronic Devices (IEDs) that are capable of publishing either one or several types of messages, as defined in this standard, shall conform to the delivery times specified in the standard. The next section provides a summary of part 6 of the IEC 61850 standard.

3.2.2 Part 6 – Substation Automation System Configuration

Part 6 of the communication standard introduces and discusses the Substation Configuration Language (SCL). The SCL is a configuration language based on the eXtensible Markup Language (XML). It enables system engineers to construct abstract models of primary and secondary substation equipment, communication mechanisms between equipment, and relationships between equipment. The Unified Modelling Language (UML) serves as the foundation for the modelling platform for this configuration language (Ncube, 2012; Apostolov, 2010).

This configuration language enables IED configuration and settings to be transmitted to a system configuration tool or another IED, decreasing the cost and labour by omitting manual intervention. Part 6 of the standard defines four different types of SCL files (Julie, 2014; Ncube, 2012; Hou & Dolezilek, 2010):

a) Documents describing the System Specific Description (SSD); Functional models define how automation systems behave when performing a function, such as protection, automation, or control.
b) Station Configuration Description (SCD); The SCD file has a fully configured communication section for IEDs or sub-systems. This file contains a list of all the IEDs located throughout the distribution substation that comprise the automation system.
c) IED Capability Description (ICD); The ICD file defines the IED's default capabilities before its name and address are configured, whereas the CID file represents a customised IED.
d) Configured IED Description (CID); The distinction between ICD and CID files is that an IED has a unique name and address.

Figure 3.2 illustrates the system configurator and IED configurator. It begins by gathering all the data from the SSD file, which includes system-linked data, and the ICD, which includes IED-linked data and then establishing the SCD file. This file creates the function and information transfer for each IED and stores it in a database. The IED Configurator then gathers the SCD file and generates the CID file. Next, the CID file is sent via a communication connection to a specific IED.

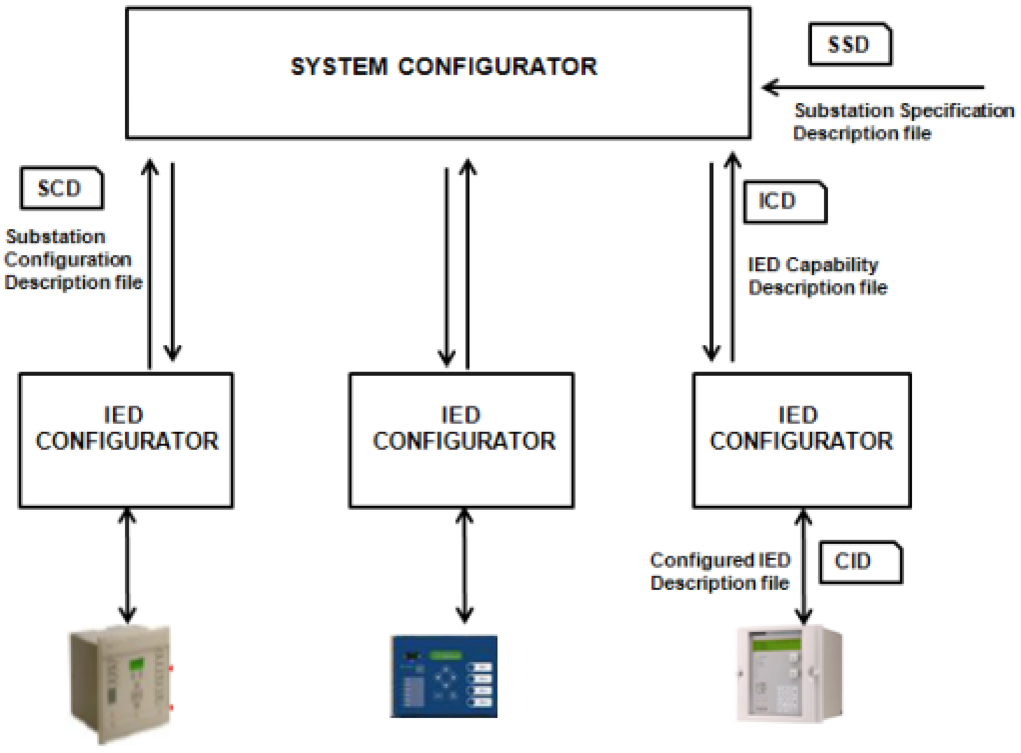

SYSTEM CONFIGURATOR
SSD
Substation Specification Description file
SCD
Substation Configuration Description file
ICD
IED Capability Description file
IED CONFIGURATOR
IED CONFIGURATOR
IED CONFIGURATOR
Configured IED Description file
CID

Figure 3.2: System Configurator and IED Configurator (Julie, 2014)

The next section summarises part 7 of the IEC 61850 standard.

3.2.3 Part 7 – Basic Communication Structure

This section of the IEC 61850 standard is subdivided into four portions, which are designated as Part 7-1, Part 7-2, Part 7-3, and Part 7-4, respectively. Providing the fundamental concepts and principles of communication amongst substation equipment in an IEC 61850 standard-based automation system is the primary focus of Part 7-1. The standard takes advantage of the virtualisation idea to provide a view of some real-world device characteristics that are critical for information exchange with other devices. Part 7-2 covers the exchange of data between entities via abstract communication service interfaces. Part 7-3 defines structures for representing common data qualities for a specific data object, and it is divided into three sections. Logical node classes and their associated common data classes are defined in Part

7-4, and these classes and data classes can be mapped to a specific communication service mapping to facilitate information sharing between entities (IEC 61850-7-1, 2003).

SASs are primarily responsible for protecting, controlling, and monitoring plant equipment employed in the substation and its feeders. These functions serve as the foundation for the IEC 61850 standard's object-oriented physical and logical device information models. The IEC 61850 standard's core is comprised of information models and methodologies for device modelling. The information models and methods for device modelling are at the heart of the IEC 61850 standard. As illustrated in Figure 3.3, the models defined in the IEC 61850 standard enable the virtualisation of physical devices. Objects, classes, attributes, inheritance, and methods are all common object-oriented concepts covered by the IEC 61850 standard. Concerning the data view and the communication view, the IEC 61850 employs object-oriented principles to describe actual physical devices and associated substation functions.

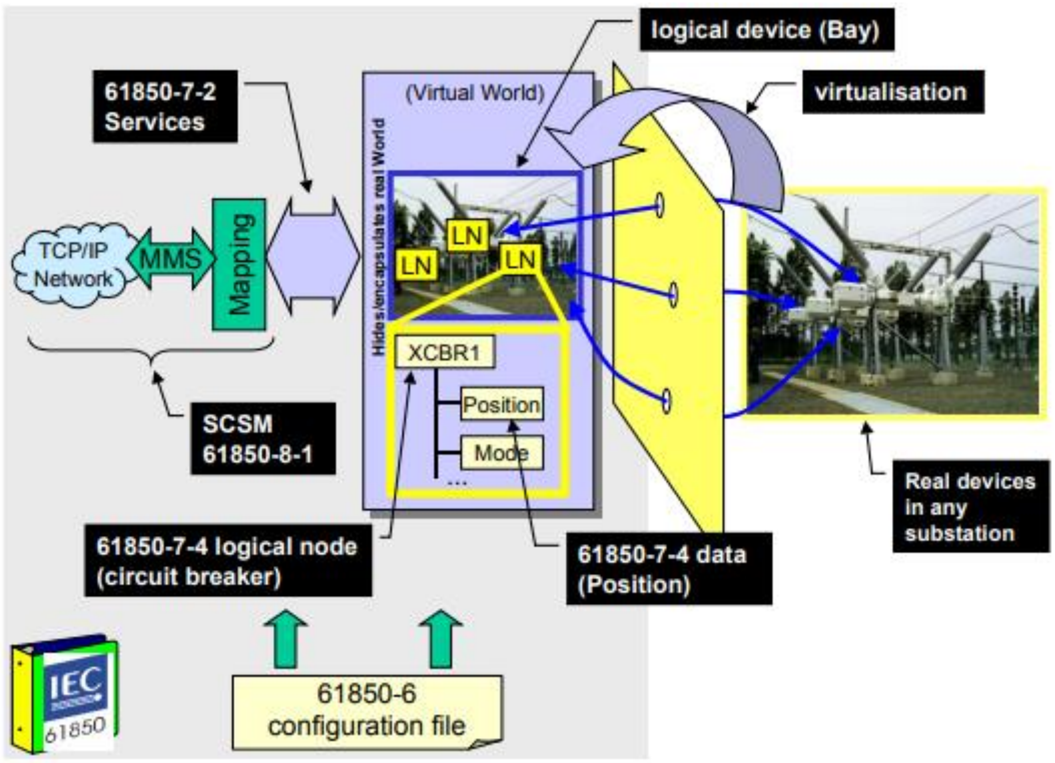

logical device (Bay)
61850-7-2 Services
(Virtual World)
virtualisation
Hides/encapsulates real World
TCP/IP Network
MMS
Mapping
LN
LN
LN
XCBR1
Position
Mode
SCSM 61850-8-1
Real devices in any substation
61850-7-4 logical node (circuit breaker)
61850-7-4 data (Position)
IEC 61850
61850-6 configuration file

Figure 3.3: Modelling approach (IEC 61850-7-1, 2003)

3.2.3.1 Abstract Communication Services

The Abstract Communication Service Interface (ACSI) is defined in Part 7-2 of the IEC 61850 standard. The standard specifies its application in substation automation when IED collaboration in real-time is necessary. The ACSI is a critical aspect in determining interoperability. Therefore, The ACSI specifies how data is exchanged between devices in a substation automation system. Moreover, ACSI is characterised as being independent of the communication systems (IEC 61850-7-2, 2003). The ACSI consists of two parts that define two different communication models: client-server and peer-to-peer architecture models as illustrated in Figure 3.4 (Kriger et al., 2013). Data modelling approaches described in parts 7-3 and 7-4 of this standard are used in conjunction with these communication services to create models (IEC 61850-7-2, 2003).

If two devices are connected in a peer-to-peer communication architecture, one is a publisher (which transmits data) and the other is a subscriber (who listens for information). Client-server architectures are frequently used for device configuration and remote access, while peer-to-peer architectures are frequently used for time-critical message exchange. IEC 61850 requires the deployment of a peer-to-peer or client-server topology to enable the ACSI, one of the critical techniques for ensuring compatibility between devices. The communication between a client and a remote server may be for data access, device control, logging of events, publisher/subscriber, file transfer, etc. (IEC 61850-7-2, 2003).

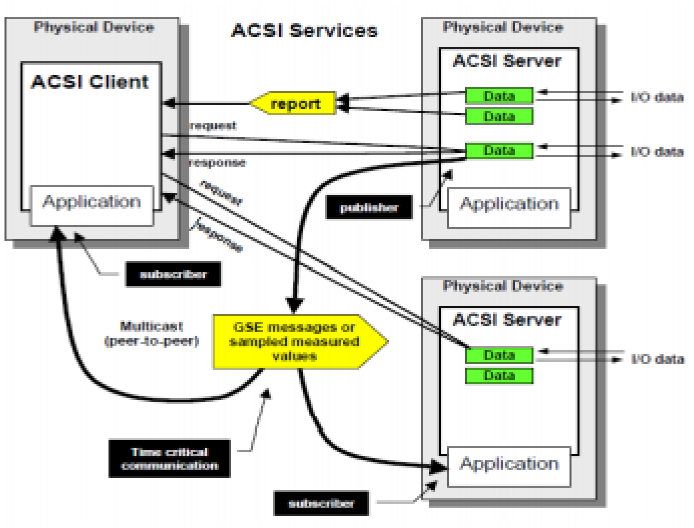


Figure 3.4: ASCI Services (IEC 61850-7-1, 2003)

To ensure compatibility, Part 9-2 of IEC 61850 specifies a Specific Communication Service Mapping (SCSM). SCSM is used to map the modelled data to well-known communication protocols such as TCP/IP, MMS, and ISO 8802-3 Ethernet frames (Mguzulwa, 2018). SCSMs describe the fundamental communication protocols and standards used between devices for data transmission using either client-server or peer-to-peer models. Parts 8 and 9 of the IEC 61850 standards define SCSM.

The next section discusses the data modelling of substation information.

3.2.3.2 Data Modelling

This is a critical part of the IEC 61850 standard since it permits interoperability across IEDs and functions. The standard is focused on describing the information models that allow the IEDs from multiple vendors to transmit information to one another.

The IEC 61850 standard makes use of data models that are representations of analogue power system equipment. These data models are generated using a process known as virtualisation. Virtualisation is the technique of presenting an automated system with a view of the elements of a physical device that are of relevance to it. Not only is virtualisation a philosophical notion that underpins the IEC 61850 modelling standard, but it is also an engineering process in which field professionals model actual substation automation equipment and operations (Retonda-Modiya, 2012).

The IEC 61850 standard permits the virtualisation of a physical power system. Figure 3.3 shows an illustration of a virtualisation process. The initial phase in the IEC 61850 modelling process that leads to the ACSI is the virtualisation of SAS components and their associated functionalities. Virtualisation is a procedure that is used to standardise the behaviour of a SAS device or function using them to facilitate communication between devices. Among the many prototypes in the IEC 61850-7 of the standard are classes of Logical Node Classes, Data Classes, and Common Data Classes. In other words, for devices and functions that have already been standardised, there is no need to go through the entire virtualisation process again.

With the use of IEDs, it is possible to reduce redundant connections of hardwiring involving RTUs, programmable logic controllers (PLCs), metering devices, and instrument transformers in today's architecture of power system automation. This requires an electrical engineer to be familiar with the IEC 61850 data modelling standard to achieve this goal.

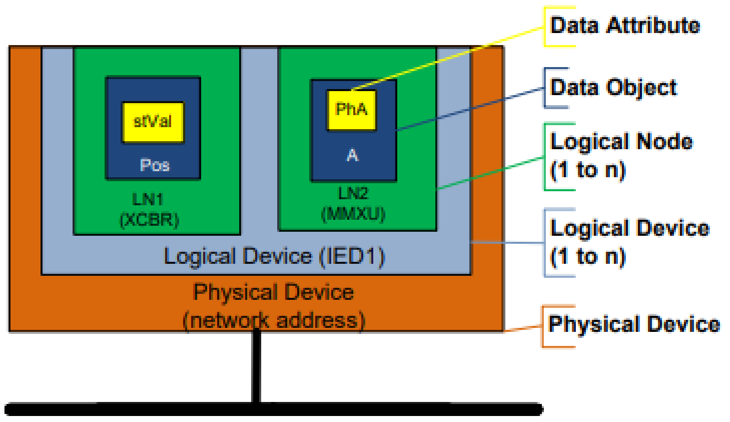


Figure 3.5: Data model layers (IEC 61850-7-1, 2003)

Data modelling in IEC 61850 begins by considering Figure 3.5 which contains five layers. Figure 3.5 depicts an object-oriented representation of the IEC 61850 standard data modelling technique. According to Figure 3.5, a physical network addressable device is comprised of a logical device, which in turn has several logical nodes for providing a variety of functionality. In Figure 3.5, the data model is overlaid with standard names, as follows (Retonda-Modiya, 2012; Mguzulwa, 2018; Emmanuel, 2014; Ncube, 2012):

a) The physical device that is directly connected to the network address is the first layer of the architecture. The real network addressable device (IED) in an automation system is identified by the physical device.
b) The second layer is the logical device, the IED. Groups of Logical Nodes that are connected within a physical device.
c) The third layer is the logical node, which represents abstract data items and the fundamental components of the IEC 61850 object-oriented virtual model. This is accomplished using standardised data and data properties. This permits the creation of a hierarchical class model in which all class information, associated services, and parameters may be accessible via a communication network.

Table 3.1 Logical Node groups in the IEC 61850-7-1 standard (IEC 61850-7-1, 2003)

Logical Node Groups	Group Designator
System Logical Nodes	L (3)
Protection functions	P (28)
Protection related functions	R (10)
Supervisory control	C (5)
Generic References	G (3)
Interfacing and Archiving	I (4)
Automatic Control	A (4)

Metering and Measurement	M (8)
Switchgear	X (2)
Instrument Transformer	T (2)
Power Transformer	Y (4)
Further power system equipment	Z (15)
Sensors	S (4)

d) The fourth layer is the data class. This is the real data that an automation system measures, monitors, or controls. Table 3.1 shows how the IEC 61850 standard classifies SAS functions into logical nodes. The groupings are further broken into 92 distinct Logical Nodes, each of which is built of 355 data classes that have application-specific meaning.

e) The data is the fifth layer. The 92 data class types are enlarged to a total of 355 data classes. A data attribute is a feature of the data object that is being monitored or measured.

The IEC 61850 standard is based on the Unified Modelling Language (UML) and employs an object-oriented modelling methodology. This configuration language makes use of UML as a basic modelling platform. After defining a class or function, the user is not required to redefine the object or function but may create an instance of that class, or object. Figure 3.6 shows the UML class diagram of the IEC 61850 data mode.

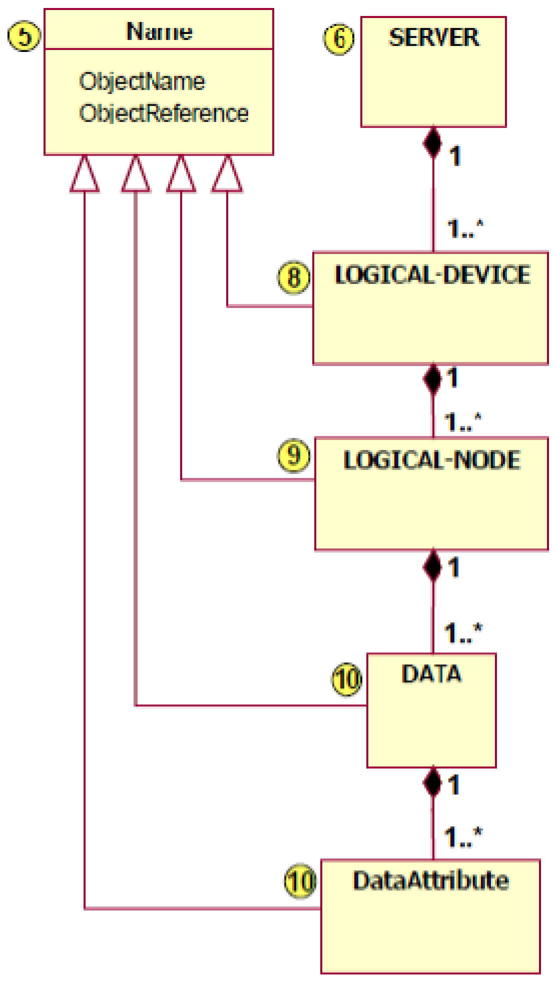


Figure 3.6: UML class diagram of the IEC 61850 data model (IEC 61850-7-2, 2003)

Part 7 of the IEC 61850 standard covers data communication models, communication services, and data classes between devices in an IEC 61850 standard-based system. The IEC 61850 standard further introduces an important factor of standard naming convention for devices, logical nodes, objects, and data attributes. However, details of the naming convention are present in this literature. Part 8-1 of the IEC 61850 standard, which defines the precise communication service mapping of application data to Manufacturing Message Specification (MMS) and ISO 8802-3 GOOSE messages, is discussed in the following section.

3.2.4 Part 8-1 – Mapping to MMS and Ethernet

It has been defined in the IEC 61850 standard that there are two types of information exchange systems. Each has its architectural framework: the first is strictly client-server communication, while the second has additional incorporates functionality based on the publisher-subscriber communication structure. To facilitate the provision of these services, the IEC 61850 standard specifies a layered communication structure on top of the OSI model.

The IEC 61850-8-1 standard defines the mapping of process level, bay level, and station level device abstract objects and services to the application layer. Along with mapping to the application layer, IEC 61850-8-1 defines profiles for the communication stack's data link, networking, transport, session, and presentation levels. ACSI and information model mappings to a specific protocol are defined in this section of the IEC 61850 standard.

Figure 3.7 illustrates the IEC 61850 standard's SCSM; defining how the IEC 61850 standard's part 7 application data is communicated utilising the SCSM. To minimise protocol overhead and message delivery times, the SCSM provides access to lower layers of the OSI model where GOOSE and SV messages are directly mapped to the data link layer, while mapping of data and information to MMS encompasses all seven of the Open System Interconnect (OSI) stack. Furthermore, the data is routed over an ethernet link layer via the ISO 8802-3 ethernet frame "EtherType" for GOOSE and SV (International Electrotechnical Commission, 2009).

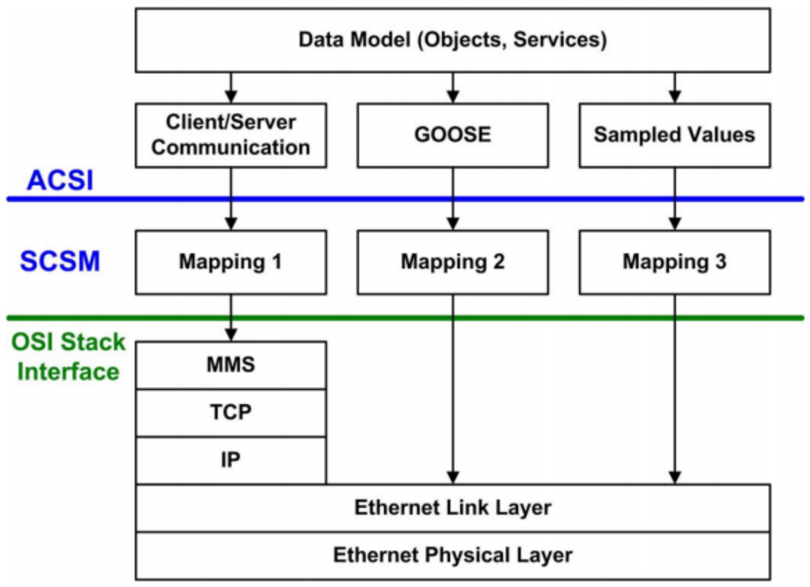


Figure 3.7: Layered structure of the IEC 61850 standard (Mohagheghi et al., 2011)

SCSMs are designated by IEC 61850-8-1, IEC 61850-9-1, and IEC 61850-9-2 standards. IEC 61850-8-1 assigns the MMS responsibility for the majority of ACSI services. Additionally, it provides mappings between Sampled Values, GOOSE, Time Synchronisation, GSSE and Ethernet. IEC 61850-8-1 defines the services that are commonly used for exchange throughout the substation. IEC 61850 defines GOOSE as the standard's core crust because it enables quick applications that meet the standard's protective performance standards.

The next section introduces the mapping of applications for the SV messages.

3.2.5 Part 9-1 and 9-2 – Sampled Measured Value Mapping.

IEC 61850 standard describes SCSM as a method of transferring current and voltage samples through a serial point-to-point link or over an Ethernet process bus to bay-level IEDs. Information is mapped across a traditional multi-drop point-to-point serial configuration in Part 9-1 of the specification; information is mapped onto an ISO 8802-3 Ethernet frame in Part 9-2 of the specification. The advantages of Ethernet switched communication, as defined in Part 9-2 of the specification, such as data speeds, zero-collision, and a flexible design, made it preferable to serial point-to-point connection. Furthermore, reducing the expense of costly and lengthy copper wiring between a substation and the control centre.

As with GOOSE messages, mapping voltage and current samples to an ISO 8802-3 Ethernet frame generates SV messages, a time-critical peer-to-peer communication protocol. There are differences between Part 9-1 of the IEC 61850 standard and Part 9-2 of the same standard not only in terms of the underlying interface but also in terms of the information that is shared. Digital inputs and voltage and current measurements are both contained in Part 9-1 of the IEC 61850 standard. Part 9-2 permits only voltage and current measurements within the defined frame, as GOOSE messages for digital status transmission are already present.

Contrary to Part 9-1 of the IEC 61850 standard, Part 9-2 restricts the transmission of SV packets without binary input statuses. The dataset is customisable in the framework of the IEC 6180-9-2 standard via the SCL. The digitisation of signals from instrument transformers is specified in Part 9-1, and the mapping of these signals into SMV frames that are transmitted via an Ethernet data link protocol is defined in Part 9-2. According to Part 9-2, an Ethernet-based network has been proposed for communication in substations to send and receive protection and automation signals between devices. The digitised signals (using Merging Units) are distributed via the Ethernet-based process bus to the bay-level protection and control IEDs. The Merging Unit is a critical component of the IEC 61850 process bus.

IEC 61850-9-2 Ethernet communication networks based on process-bus can provide numerous benefits, including a flexible architecture, and interoperability. For time-critical messages, such as SMV, it is crucially important that the feasibility and reliability of substation protection systems operating over an Ethernet-switched Process-Bus network be thoroughly investigated, even though the Part 9-2-based process bus has proven to be a promising technology for protection systems (Abdolkhalig, 2014).

3.3 Application of the IEC 61850 standard in a Substation Automation System

As utilities worldwide embrace the IEC 61850 digital substation standard, completely automated techniques of power system monitoring will eventually replace traditional strategies. While emerging techniques improve efficiency and controllability, they require thorough evaluation and analysis before being widely used.

Standardisation of communication systems in a distribution substation automation system has been achieved through the introduction of the IEC 61850 standard. Throughout history, communication has always been important in the operation and automation of the power system. As a result, the communication standard was created to interface with modern technology and provide features that were previously unavailable through legacy communications protocols. When compared to a legacy method, IEC 61850 allows for advances in the power system that would otherwise be impossible. These distinguishing qualities of the new communication standard for IEDs have a significant effect on the costs associated with the design, construction, installation, commissioning, and operation of power systems, among other things. As described previously, the process bus is composed of two real-time data transfer protocols: the GOOSE and SV messages defined in Parts 8-1 and 9-2, respectively. Per the IEC 61850 standard, a SAS is divided into three distinct levels, which are represented by stations, bay, and process levels, as indicated in Figure 3.8.

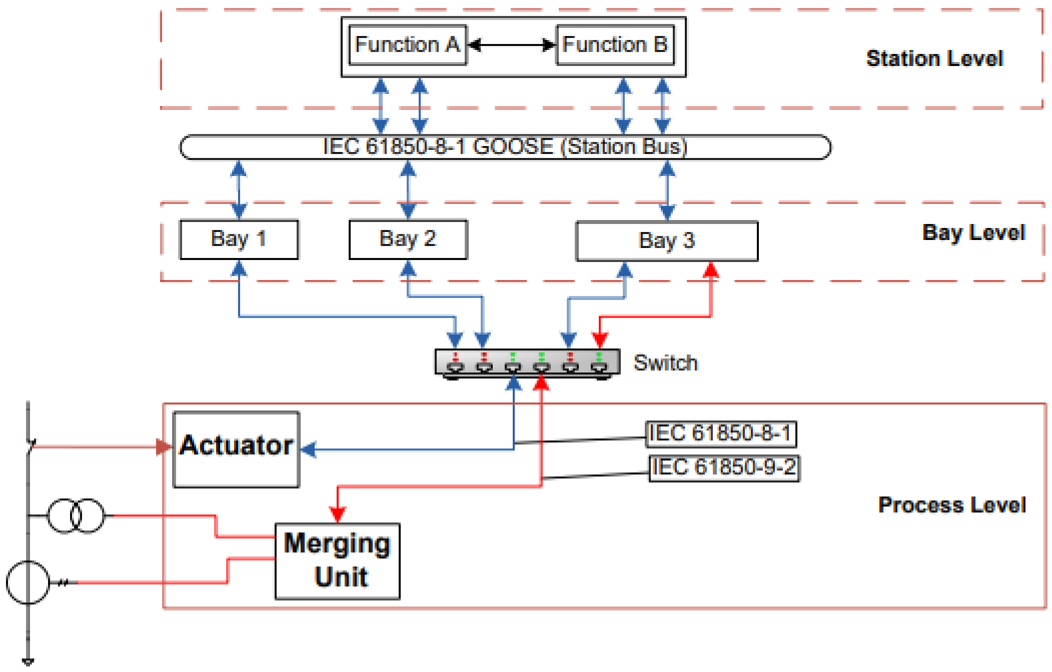


Figure 3.8: IEC 61850 System Architecture (Zhao, 2012)

The introduction of the IEC 61850 standard defined communication for substation automation systems. The introduction of the IEC 61850 standard has brought about several benefits for the Substation Automation System (SAS) environment. The requirement for interoperability among relevant devices from various vendors is a requirement for ensuring that utilities derive value from their investments. Interoperability, according to the IEC 61850 systems, is a significant advantage in the Substation Automation System (SAS) context (Mguzulwa, 2018). To successfully implement interoperable systems in the SAS environment, thorough testing and a careful selection of vendors are required. This entails thorough testing to ensure that the SAS meets the standards of a certain SAS. Interoperability implementation and testing techniques in a SAS must be created and extensively validated with multiple situations of interoperability to be considered successful (Mguzulwa, 2018).

The subsequent sections, 3.3.1 and 3.3.2, elaborate on the GOOSE and SV message structures, respectively.

3.3.1 IEC 61850-8-1 standard GOOSE Messages

GOOSE messaging refers to the time-critical messaging sent between IEDs in a peer-to-peer fashion; these messages can be multicast to many IEDs or directed to a single IED via a unicast address. In general, GOOSE messages are used to exchange status information between two or more devices, notably the publisher and the numerous subscribing devices; for example, circuit breaker status and voltage measurements are shared between functions in a protective scheme. GOOSE messages are continuously transmitted via the Local Area Network (LAN). Figure 3.9 depicts an illustration of the publisher-subscriber method that is utilised in the GOOSE message exchange as an example.

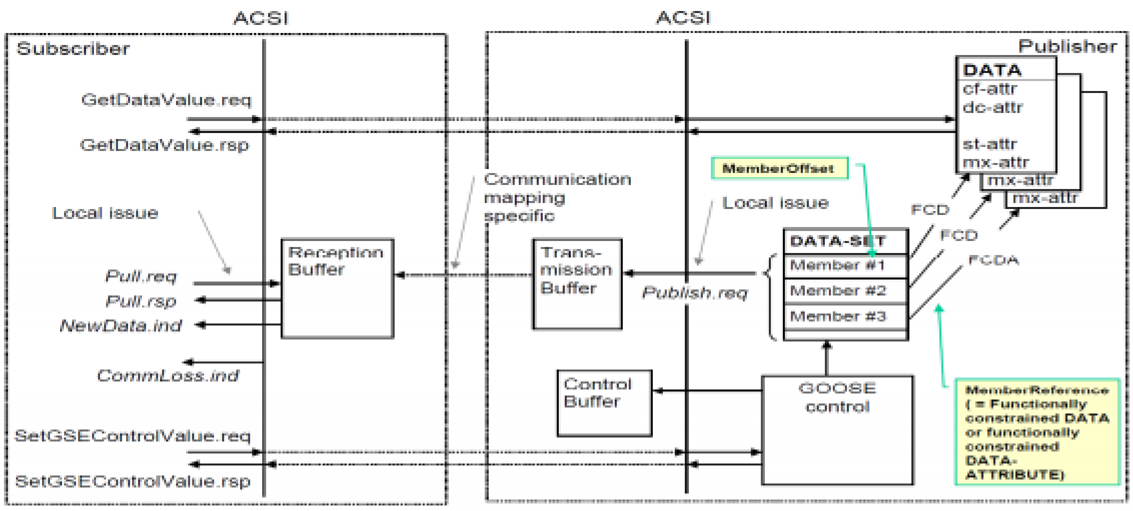
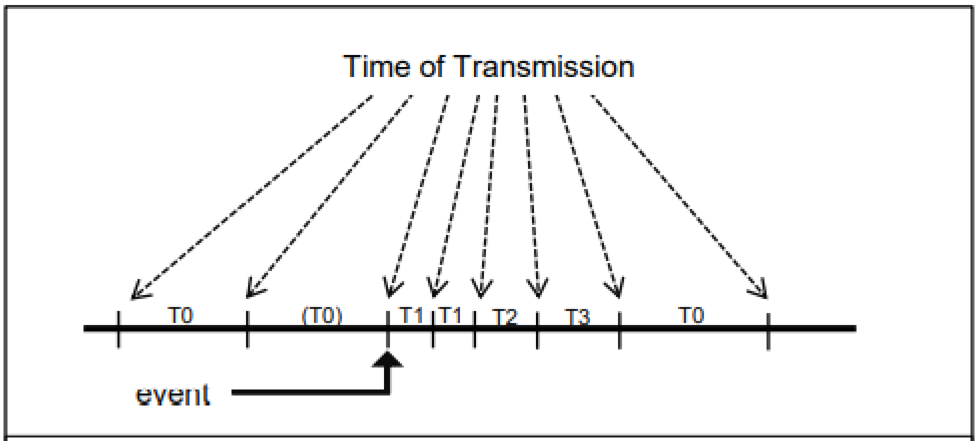


Figure 3.9: An example of the GOOSE Publisher-Subscriber mechanism (IEC 61850-7-2, 2003)

It is critical to achieve or maintain interoperability when GOOSE messages are sent. Data transmission benefits provided by IEC 61850 are particularly advantageous for time-critical data exchange between functions located within a bay or across many bays. GOOSE communications rely on peer-to-peer communication to address the needs of those who require protection. (Mguzulwa, 2018)

The goal of GOOSE messaging is to maintain the state of logic or analogue values in IEDs indefinitely. Continuous indication via control messaging is critical if the GOOSE messaging concept is to be achieved. The subscriber is defined as any IED equipped to consume the message's content, whereas the publisher is defined as the IED that broadcasts the message. When the data in the dataset is modified, the publisher's transmission buffer is changed, and the values are sent through a GOOSE message. As a result, the publishing device floods the LAN with the latest GOOSE messages, and the subscriber must guarantee that they are captured. The GOOSE messages provide information that informs the subscriber when a status has changed and at what time it occurred. Every few milliseconds, IEDs must update their contact status and analogue values. This means that each publication will repeat the most recent messages numerous times to keep subscribers informed. GOOSE messages are conveyed in a continuous fashion using a repetition approach as illustrated in Figure 3.10. To ensure that GOOSE messages are delivered at the desired speed and reliability, IEC 61850-8-1 standard provides a retransmission mechanism for GOOSE messages. (Retonda-Modiya, 2012; Mguzulwa, 2018; Emmanuel, 2014; Ncube, 2012; Abdolkhalig, 2014)



T0	retransmission in stable conditions (*MaxTime)*
(T0)	retransmission in stable conditions may be shortened by an event
T1	shortest retransmission time after the event (*MinTime*)
T2, T3	retransmission times until achieving the stable conditions time

Figure 3.10: Repetitive transmission of GOOSE messages

GOOSE is a connectionless, event-driven communication service. After a certain interval, a GOOSE client transmits GOOSE messages continually (T0). When a substation event occurs, i.e., when the status value of one or more data items in the GOOSE dataset changes, the GOOSE transmission interval drops dramatically, as seen in Figure 3.10 (T1). Following the occurrence, the transmission settles into a predetermined interval, which remains constant until the next event happens (T2, T3, and T0). The GOOSE message retransmission mechanism is necessary to transmit messages from the publisher to the subscriber device for the subscribing device to verify that communications via process bus are healthy (Abdolkhalig, 2014).

The GOOSE Control Block (GoCB) manages the GOOSE message services and database. Control blocks specify the rate at which data is transferred between IEDs and the method by which it is communicated using abstract communication services. As seen in Table 3.2, the GoCB specifies information such as the GoCB name, the GoCB reference, the GOOSE enable, the dataset, and other attributes that permit the delivery of a GOOSE message.

Table 3.2: GOOSE Control Block Class (International Electrotechnical Commission, 2009)

GoCB Class				
Attribute name	**Attribute type**	**FC**	**TrgOp**	**Value/Value Range**
GoCBName	ObjectName	GO	-	Instance name of an instance of GoCB
GoCBRef	ObjectReference	GO	-	Path-name of an instance of GoCB
GoEna	BOOLEAN	GO	dchg	Enabled (TRUE) | disabled (FALSE)
AppID	VISIBLE STRING65	GO	-	System-wide identification
DatSet	ObjectReference	GO	dchg	
ConfRev	INT32U	GO	dchg	
NdsCom	BOOLEAN	GO	dchg	
Services SendGOOSEMessage GetGoReference GetGOOSEElementNumber GetGoCBValues SetGoCBValues				

A dataset defines the message content. An ethernet frame encapsulates the GOOSE message. The ethernet frame's header contains information about the destination, source, ethertype, and payload. IEC 61850 defines the payload's structure.

The next section explains the GOOSE message structure as established in the IEC 61850-8-1 standard.

3.3.1.1 GOOSE Message Structure

The compatibility of GOOSE messages must be tested consistently to guarantee that IEDs are adequately validated. IEC 61850 is a comprehensive standard, and compliance with this standard does not always indicate interoperability with equipment from different suppliers. The interoperability of GOOSE communications enables a variety of applications in an automation substation. GOOSE messaging is essentially peer-to-peer messaging that may be used to construct any application that requires a message. When compared to hardwired alternatives, the use of the GOOSE message has been shown to reduce fault-clearing time. While compliance tests may have been performed, prior research indicates that message structures are critical for interoperability. (Mguzulwa, 2018) state advantages of GOOSE message namely:

- GOOSE protocol uses the standard Ethernet for communication.
- Ethernet replaces point-to-point hardwired copper.
- Speed performance requirements are improved.
- Due to the message's periodic repetition, it has a high degree of reliability.
- The GOOSE is directly mapped to the Ethernet layer hence the processing time is also less which is best suited for very time-critical protection functions in the substation.

The benefit of GOOSE messages is critically significant in the deployment of interoperability because these advantages must be realised by a variety of suppliers in the substation automation system to be effective.

GOOSE messages are mapped to the ISO 8802-3 Ethernet frame, and the Protocol Data Unit (PDU) is included in the Ethernet frame's data payload part. The ISO 8802-3 frame consists of the following: (International Electrotechnical Commission, 2009)

- **MAC Destination address:** This is the MAC address of the destination device. The destination address is defined as a multicast address.
- **MAC Source Address:** This is the MAC address of the GOOSE sending device.
- **VLAN Tag:** GOOSE frames are tagged using the IEEE 802.1Q to separate time-critical messages from low-priority data. The Priority tagged field consists of three fields namely: Tag Protocol Identifier (TPID), Tag Control Information (TCI), and Virtual LAN Identifier (VLID).
- **EtherType information:** The Ethernet PDU consists of four fields namely: Application Identifier (APPID), Ethernet type, Length, and GOOSE Application Protocol Data Unit (APDU).

In multicast addressing, the destination device's MAC address should be in the range 01-0C-CD-01-00-00 to 01-0C-CD-01-01-FF. The VLAN field's TPID is set to 0x8100 to differentiate IEEE 802.1Q-tagged frames from untagged frames. For GOOSE messages, the EtherType is 0x88B8, and the APPID is set to 0x0000 by default. The IEC 61850-8-1 standard specifies that the GOOSE APPID may be any value between 0x0000 and 0x3FFF. (International Electrotechnical Commission, 2009)

The structure of the ISO 8802-3 Ethernet frame is shown in Figure 3.11.

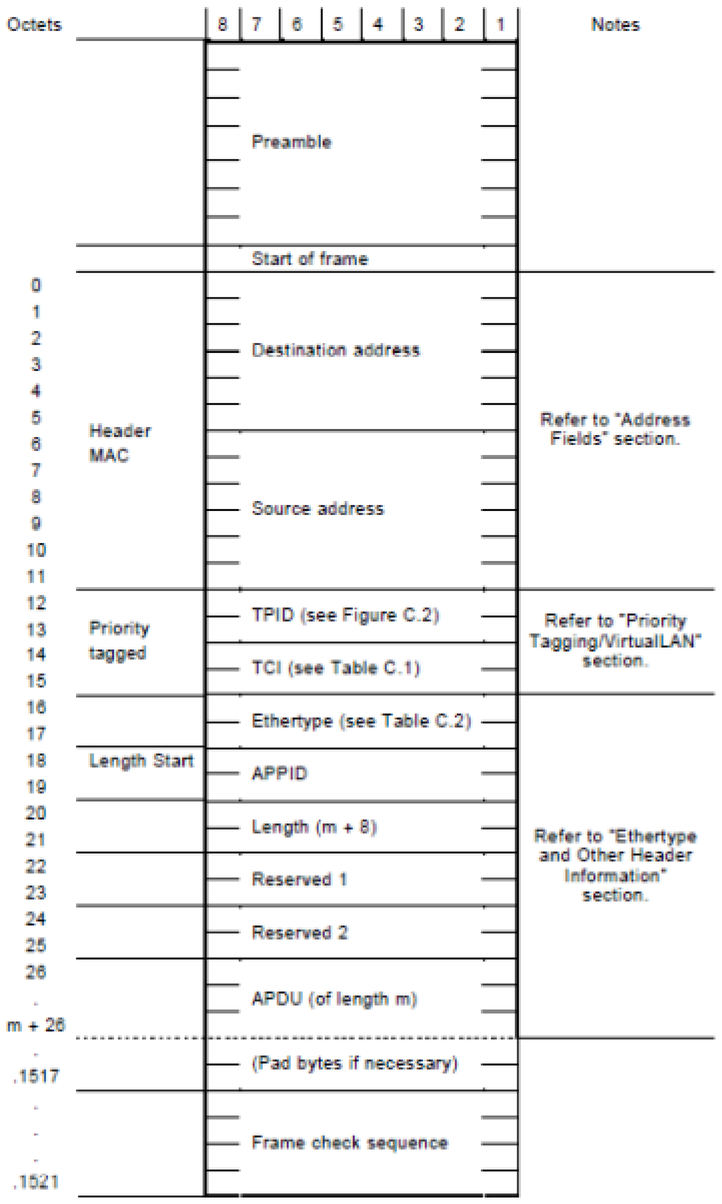


Figure 3.11: GOOSE Message Frame Structure (International Electrotechnical Commission, 2009)

The GOOSE APDU consists of the following fields: (Hariri et al., 2019; Ncube, 2012; Emmanuel, 2014; Abdolkhalig, 2014; Gadelha Da Silveira & Franco, 2019)

- **State number (stNum):** This field contains a state number of the client's state machine that indicates when an event occurred, namely when a GOOSE message changed state or when an event occurred.
- **Sequence number (sqNum):** This field keeps track of the number of GOOSE messages transmitted over the network. This value is incremented once an event

happens until the next event occurs or the retransmission time reaches the stable retransmission time. When the next event occurs, the value turns to zero.

- **Simulation/Test:** This field provides insights into whether the messages published are from a test operation or live scenario.
- **Time allowed to live (TAL):** This field contains information about the maximum amount of time that the packet has to travel to its destination.
- **Need to commission (NdsCom):** This field specifies whether or not the GOOSE client/publisher requires commissioning; and validating the GOOSE message.
- **Configuration revision (confRev):** This field specifies the configuration file's revision number at the time of transmission. This value is increased whenever the dataset's data items are reordered, deleted, or added.
- **Number of dataset entries (numDatSetEntries):** The value in this field shows the number of data sets.
- **GOOSE control block reference (gocbRef):** A unique reference to the control block associated with the GOOSE message.
- **Data set (datSet):** The GOOSE dataset name is specified.
- **GOOSE ID (goID):** The field indicates the name of the GOOSE dataset that sends the message.
- **Timestamp (t):** The field indicates the time when the GOOSE message is generated.
- **Data:** The field indicates the information of the GOOSE message (bool, integer, float…).

Publisher IEDs encrypt the information contained in a data set and build an envelope for it to be delivered in a package. When this packet is received by subscriber IEDs, the information included in the gocbRef, datSet, goID, confRev, and numDatSetEntries parameters is used to validate and process the messages. After the message has been validated, the information included in the data set is used to carry out the IED logic.

The GOOSE message structure is discussed. The next section discusses the SV message structure defined in the IEC 61850-9-2 standard.

3.3.2 IEC 61850-9-2 SV messages according to IEC 61850-9-2LE

The IEC 61850-9-2 standard defines the process level as a means of collecting data, such as voltage and current, from transducers attached to the major power system equipment. As defined in the IEC 61850 standard, SV messages are linked to measurement dispersion. They are moved from the bay to the process level of the power system. Time limitations apply to the transmission of sampled values. It is possible to transmit sampled values in an ordered and time-controlled way using the IEC 61850 protocol architecture by mapping SV messages directly to the data link layer of the ethernet network stack. Time-critical SV messages can be exchanged between two or more devices periodically. The SV data exchange has a publisher/subscriber model as illustrated in Figure 3.12.

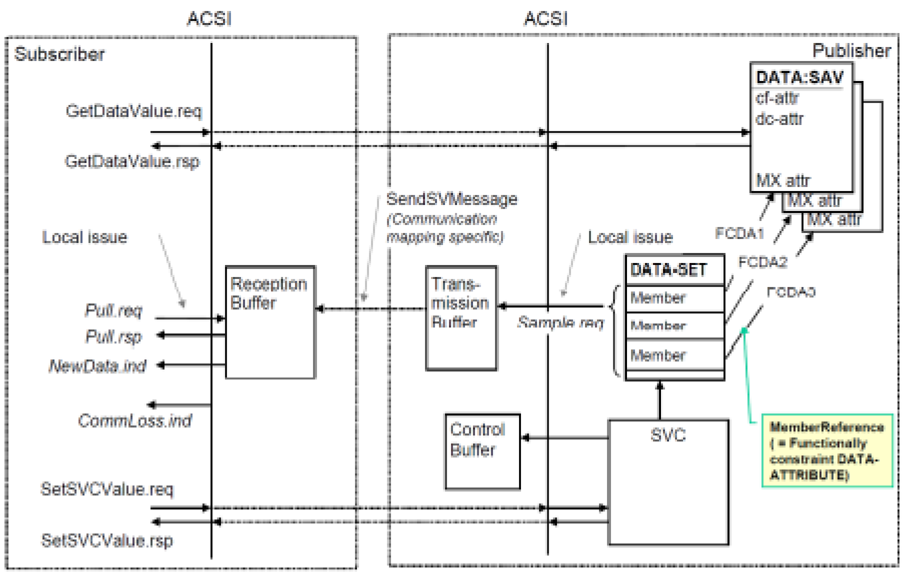


Figure 3.12: Sampled Value Control Class (IEC 61850-7-2, 2003)

On the sending side, the publisher stores the data in a local buffer, and the subscriber gets the values from the receiving side's local buffer. The count of sampled values is appended as a timestamp, allowing the subscriber to check the values' timeliness. The multicasting of time-critical messages should be supported by any IED that communicates via the IEC 61850-9-2 process bus. Multicasting allows for the distribution of the same message to numerous subscribers at the same time.

The Sampled Value Control Block (SVCB) manages the SV message services and database. The SV services are used to manage the communication process between the publisher and the subscriber. As seen in Table 3.3, the SV control block is detailed.

Table 3.3: SV Control Block Model (IEC 61850-7-2, 2003)

MSVCB Class					
Attribute name		**Attribute type**	**FC**	**TrgOp**	**Value/Value Range**
MsvCBNam		ObjectName	-	-	Instance name of an instance of MSVCB
MsvCBRef		ObjectReference	-	-	Path-name of an instance of MSVCB
SvEna		BOOLEAN	MS	dchg	Enabled (TRUE) | disabled (FALSE)
MsvID		VISIBLE STRING65	MS	-	
DatSet		ObjectReference	MS	dchg	
ConfRev		INT32U	MS	dchg	
SmpRate		INT16U	MS	-	(0…MAX)
OptFlds		PACKED LIST	MS	dchg	
	refresh-time	BOOLEAN			
	sampled-synchronised	BOOLEAN			
	sample-rate	BOOLEAN			
Services SendMSVMessage GetMSVCBValues SetMSVCBValues					

IEC 61850 messages are directly connected to the ethernet stack's data link layer. The User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) transport layers are no longer used since they introduce additional processing latency as a result of the ethernet frame headers. The performance of both SV and GOOSE communications is critical since the power network's proper operation is dependable on their timely transmission. In switched Ethernet networks, there are numerous sources of delay, each of which contributes to the total performance. Additionally, lengthening the packets causes the message to consume more bandwidth, putting the network under pressure. Bandwidth has a significant impact on network delay. (Skoff, 2020)

Because protection systems maintain the grid's reliability, their performance is critical. As IEC 61850 becomes more widely implemented in power systems worldwide, sampling value-based methods will unavoidably become the industry standard for identifying and isolating problems. As a result, they must undergo thorough testing to ensure that they can be relied upon in the same manner that traditional protection systems are. (Skoff, 2020)

The next section explains the SV message structure as established in the IEC 61850-9-2LE standard.

3.3.2.1 IEC 61850-9-2 SV structure and IEC 61850-9-2LE data model

IEC 61850 Part 9-2 specifies a user-configurable SV message frame, which may be customised using the substation configuration language. The implementation of the IEC 61850-9-2 communication service mapping is simplified by the definition of datasets, ASCI, physical connections, and sample rates. This section discusses message mapping using sampled values, as defined in IEC 61850-9-2LE. As demonstrated in Figure 3.13, IEC 61850-9-2LE SV messages are translated to an ISO 8802-3 Ethernet frame that consists of a header and the SV APDU.

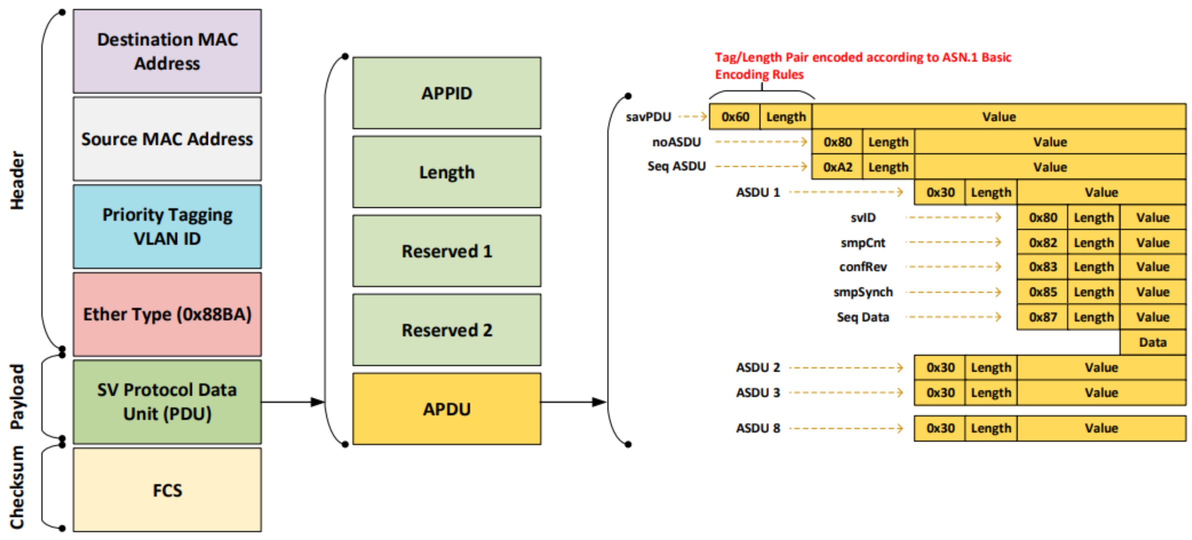


Figure 3.13: Structure of an IEC 61850-9-2 SV Message (Hariri et al., 2019)

Figure 3.13 shows the structure of an SV message. It is composed of three parts, namely a header, a payload, and a checksum. The Ethernet frame header contains the following information: (Hariri et al., 2019; Ncube, 2012; Emmanuel, 2014; Abdolkhalig, 2014; International Electrotechnical Commission, 2009)

- **MAC Destination address**: IEC 61850-9-2 employs Media Access Controller (MAC) filtering to ensure that SV messages are delivered on time. As detailed in IEC 61850-9-2, the target address in multicast mode should be in the range 01-0C-CD-04-00-00 to 01-0C-CD04-01-FF.
- **MAC Source Address:** This is the MAC address of the source device.

- **VLAN Tag:** SV frames are tagged using the IEEE 802.1Q to separate time-critical messages from low-priority data. The Priority tagged field consists of two fields namely: TPID and TCI. The TCI field is further divided into Canonical Format Identifier (CFI) and VLID. For SV frames, the TPID should be set to 0x8100. When set to 0x8100, the frame is identified as an IEEE 802.1Q frame.
- **EtherType information:** This field is used to specify which protocol is contained within an Ethernet Frame's Payload. The EtherType for IEC 61850-9-2 SV packets is 0x88BA. APPID is used to distinguish between Ethernet frames containing SV messages. APPID identifies the network information and is set to 0x4000 for SV messages.
- **SV Application Protocol Data Unit (SV APDU):** This is the sampled value APDU that contains the application data. The SV APDU in Figure 3.13 consists of one or more Application Service Data Units (ASDUs). These fields are the svID, smpCnt, confRev, smpSynch, and the sequence of data. Each ASDU then contains seven subfields which are as follows:
    - **Sampled Value ID (svID):** This field is a unique identification of the sampled value buffer.
    - **Sample Count (smpCnt):** This is an incremental counter that increases each time a new sample value is received. The smpCnt parameter is equal to the number of published SV messages each cycle.
    - **Configuration revision (confRev):** This field specifies the value that indicates the number of configuration changes. This value is increased whenever the dataset's data items are reordered, deleted, or added.
    - **Sampled Synchronised (smpSynch):** A Boolean value that indicates whether the SV is synchronised with a clock signal or not by setting it to either TRUE or FALSE.
    - **Sequence of data (Seq Data) and dataset:** This field contains the dataset to be transmitted in the SV packet.
    - **Refresh Time (RefrTm):** This contains the refresh time of the SV buffer.

In a sampled value digital substation, an IEC 61850-9-2LE compliant device would publish SV streams of four currents (IA, IB, IC, and IN) followed by four voltages (VA, VB, VC, and VN). The IEC 61850-9-2LE recommends a publication rate of 80 or 256 samples per cycle for SV messages on 50Hz and 60Hz systems, respectively. Violation of these rates for a specific sampled value stream will enable the detection of a DoS attack. Synchronisation is critical in an SV substation because SV streams generated by many physical devices cannot be reliably interpreted without synchronised time sampling. This guarantees the time coherence of all devices, facilitating precise data analysis and decision-making.

The IEC 61850-9-2LE provides a pre-defined data model and dataset for the transmission of current and voltage samples. This data model contains four TVTR and four TCTR logical nodes representing voltage and current transformers respectively. Each instance of the current transformer (TCTR) and voltage transformer (TVTR) logical nodes digitises one phase or neutral current or voltage. The primary current and voltage sample values are members of the dataset PhsMeas1 used in the ASDU. Figure 3.14 illustrates the data model for IEC 61850-9-2LE SV messages.

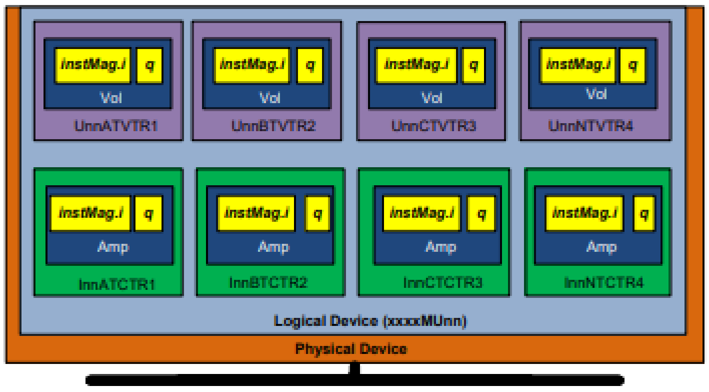


Figure 3.14: IEC 61850-9-2LE data model

The TVTR and TCTR logical node classes, as depicted above, contain the data objects for the voltage and current samples. To decrease implementation complexity, IEC 61850-9-2LE makes use of an instance of the SAV common data class that only supports characteristics with the measurement (MX) functional constraint. Each sampled value published in an IEC 61850-9-2LE contains the instantaneous magnitude (instMag.i) value of the analogue voltage or current and the quality attribute (q) which contains flags that are set by the source to inform the receiving device about the validity and other quality-related issues of the sample.

In analog GOOSE, the Deadband (db) is a range surrounding a particular measurement value that does not trigger any reporting or communication. This is done to avoid excessive communication and reporting of little, unimportant changes in analog values. Instantaneous magnitude is a parameter in analog GOOSE that specifies the current value of an analog quantity at any given time. The Deadband parameter is linked to instantaneous magnitude in that it specifies the range surrounding the current value within which no reporting or communication will occur. If the change in the analog value falls within the Deadband, it is deemed inconsequential, and the system generates no new message to communicate the change.

The following section summarises the benefits of adopting the IEC 61850 standard for communication networks.

3.4 Security Risks and Challenges in IEC 61850

In addition to making the smart grid far more efficient and sustainable in addressing the expanding global energy concerns, the growth of cyber-physical entities has also introduced numerous vulnerabilities that have led to breaches in data integrity, confidentiality, and availability. Power control grids based on IEC 61850 are being given careful consideration concerning information security risks and hazards. Using IEC 61850 in a similar setting is the ideal situation for secure IEC 61850-based substations. IEC 62351 is not supported by all currently manufactured IEDs from various manufacturers. Thus, other security measures like Intrusion Detection Systems (IDS) are required to guard against potential assaults on the current IEC 61850-based networks. Smart grids' security is extremely important due to the anticipated heavy reliance on IEDs and cyberattacks could result in substantial technical and financial losses. Analysing the smart grid's vulnerabilities and identifying mitigation strategies are essential. If cybersecurity vulnerabilities are not examined, the physical power system may be compromised, and operations may fail. (Elgargouri & Elmusrati, 2017)

According to (Elgargouri & Elmusrati, 2017), the following are potential attacks on IEDs in the present IEC 61850 local area networks:

1. Unauthorised Access
2. Denial of service
3. Spoofing
4. Data interception
5. Man-in-the-middle attack
6. Configuration Tampering
7. Operation System Attack

The aforementioned attacks may have a variety of effects on the smart substation, causing various forms of network damage. The following are the most prevalent effects of various cyber-attacks:

1. Denial of service from the control system
2. Interruption of protection communication
3. Interruption of the monitoring system
4. Network interruption
5. Protection tripping failure

These effects pose substantial risks to both networks and human life. This means that, even though these attacks are rare, their impact must be considered, and prevention strategies established. IEC 61850 features various security flaws that competent attackers could exploit to compromise the system, perhaps resulting in a blackout. When implementing IEC 61850, the following major aspects should be examined (Massink, 2016):

1. Hardcoded functions: IEC 61850 offers powerful functionalities that can result in unforeseen events. This makes access control levels extremely difficult and limits the device's security protection.
2. Authentication: The IEC61850 MMS-based protocol includes authentication. Nevertheless, the approach is not widely supported and employs plain-text passwords.
3. Key management: Key management provides additional risk, such as specially developed key management infrastructure that does not address the relevant issues, which might expose a system while providing a false impression of security.
4. Firmware integrity: Typically, firmware is not signed, and there is no method to verify its integrity. This could enable certain advanced assault scenarios, especially if the supply chain is uncontrolled.
5. Message Integrity: The GOOSE protocol does not provide publisher authentication. This means that anyone on the network can pose as a publisher. Although various efforts have been made to secure GOOSE by inserting a signature, it was demonstrated in 2010 that the time and performance requirements of GOOSE for protection methods currently make it technically difficult to construct a satisfactory solution using the existing specification.

Any unencrypted data is vulnerable when using the IEC 61850 protocol for substation automation communications outside the substation. If firewalls or data gateways are not used, substation LANs are exposed. All data is exposed if the utility depends on third parties for outside communications. Do security risk analyses of every data entering or exiting the substation to ascertain whether encryption is practical and, at the very least, authentication. If a cyber incursion is involved, the price of inaction may be incalculable. Before choosing a supplier to provide smart substation devices, exercise prudence and extensive testing. Before applying to the live system, all security updates and patches should be tested in a lab or testing environment. In the event of a suspected vulnerability or threat, make sure providers or vendors are promptly updating customers.

3.5 Security Requirements for IEC 61850 Messages

Due to the significant growth of unmanned substations and renewable energy sources like solar power and wind turbines, power grids have become significantly more complex. In terms of power generation, renewable energies have so far had the biggest impact on power networks, resulting in an unstable grid.

This makes it challenging to control the real-time distribution of power. Digital substations, which offer stability and flexibility regarding power supply, are becoming important and crucial in power transmissions to address these instabilities in power networks. Therefore, ethernet-based devices must be used instead of traditional serial-based devices, unless the serial-based devices can communicate on an ethernet-based network. But another problem emerges: since ethernet-based networks are constantly at risk from cyberattacks, as a result of this development, energy networks are now vulnerable to the same vulnerabilities that IT-based systems are. IEC 61850 was not initially built with security in mind, which may provide a significant risk if security is not promptly addressed. Cybersecurity implementation must be prioritised in power substations. As a result of an unsecured IEC 61850-based communication system, cyber hackers may acquire access and establish communications within a network. As a result, methods to detect assaults are required to respond quickly and minimise the damage.

While adopting IEC 61850, (Massink, 2016) advises considering the following factors:

1. It is important to confirm that IEC 61850 is limited to the substation's local network and is unable to access any other networks.
2. Prohibit unlawful communications via a firewall at the substation level by employing the Deny All rules approach.
3. Observe the substation network for unusual activity by using packet filtering firewalls, and utilising proper Intrusion Detection System (IDS) solutions.
4. Restrict third-party access to the substation by implementing network segmentation, demilitarized zones, deploying firewalls and using Virtual Private Networks (VPNs).

5. Establish appropriate security specifications for vendors by establishing compliance frameworks such as North American Electric Reliability Corporation Critical Infrastructure Protection (NERC CIP), ISO 27001 Certification, NIST frameworks or MITRE ATT&CK.
6. Establish a reliable security management approach by utilising proper Security Information and Event Management (SIEM) solutions.
7. Assess the security of devices to detect potential vulnerabilities and provide mitigation measures by using assessment tools like the Common Vulnerability Scoring System (CVSS).

Critical power activities are managed by substation protection and control systems using communication protocols such as IEC 61850. Communication protocols rarely have sufficient security measures, while playing a very crucial function. Because of this, there is a significant possibility that rogue attackers could access old IEDs through these communication protocols and disrupt systems causing enormous financial damages. For industrial automation and control systems, the IEC 62351 standard has grown to be one of the most widely used cybersecurity standards. Operators must take into account techniques to safeguard sensitive data and monitor the state of network security when using the recognised technologies.

To stop these types of malicious activities, the following measures must be implemented (Moxa, 2022):

1. Communication gateways such as ethernet switches or converters must encrypt data for communication protocols such as IEC 61850 and program tampering resistance. To reduce security attacks, critical data must be encrypted.
2. Monitoring the security status of network devices and any malicious activities. Making the network infrastructure more secure is essential.
3. Install IEC 61850-compliant communication gateways to secure devices and increase communication security.
4. Configure devices securely with security-embedded functions to ensure the device is secure in the initial configuration process. DDoS defence with integrated suspicious activity detection capabilities
5. Use strong passwords to avoid unwanted access.
6. Implement sniffer and data breach protection.

7. Protocol encryptions increase communication security. If communication is not encrypted, hackers may learn how to operate your IEDs and then give them a bogus control command, jeopardising the operation of your substation.
8. Installation of a communication gateway that conforms to IEC 61850 security features to offer a comprehensive cybersecurity package featuring intrusion prevention systems.

A study on "how standards improve substation cybersecurity" is presented by (Steinhauser & Klien, 2021). The authors offer suggestions for enhancing substation security. It's crucial to remember that newly installed substations use modern technologies and standards, which significantly enhance cybersecurity. The National Institute of Standards and Technology (NIST), which employs the security functions of *“Identify, Protect, Detect, Respond, and Recover,”* is the first standard the authors suggest. The objective of the standard is to enhance the security of critical infrastructure. Additionally, the authors advise using IEC 61850 devices that provide intrusion detection and implementing cyber risk management in IEC 61850 devices to report actual threats. The intrusion detection system can track the exchange of information between IEDs and assess network packets in comparison to actual data models. Recently, the authentication method IEC 62351 was introduced to secure GOOSE and R-SV transmissions. The subscriber device can Recognise and disregard bogus IEC 61850 communications by implementing IEC 62351. The cybersecurity of communication networks used by power utilities is considerably aided by the use of standardised protocols and techniques. Consequently, IEC 61850 can be more effective, dependable, and secure than conventional communication techniques in substations, making it a preferable choice. The security challenge of IEC 61850 communication and transmission is handled to some extent, and protocol encryption is implemented.

3.6 Overview of IEC 62351

The current standard for data transmission and security in digital substation systems is IEC 62351. It focuses on the key criteria for secure data transmission and communication, such as data integrity, secrecy, and authentication. The introduction of IEC 62351 addressed a challenge in securing power systems by bringing traditional non-secure communication methods up to speed. The requirement for cyber security and the implementation of IEC 61850 have been major discussions in the electrical environment.

End-to-end security for smart grids can be attained by using the IEC 62351 security standard to protocols like IEC 61850 and traditional protocols such as DNP3, and 60870-5-101/-104.

The standard series stipulates the importance of authentication and authorisation as well as encryption to ensure integrity. IEC 62351-3, for illustration, defines Transport Layer Security (TLS) encryption. For serial protocols and devices that are unable to fulfil the computational demands of encryption, IEC 62351-5 offers various options. It cannot provide data secrecy without encryption. To enable authentication for peer-to-peer multicast protocols like R-SV, IEC 62351-6 offers a method for digitally signing messages. IEC 62351-6 standard defines security protocols in IEC 61850 such as GOOSE and SV. To avoid interfering with the system's smooth operation, the security mechanisms contained in these messages must introduce the least amount of latency possible. This criterion had an impact on the final version of the standard, which calls for required authentication and message integrity. Encryption-based confidentiality has been designated as an optional feature. These signals (IEC 61850 protocols) must be sent within 3 ms, thus encryption and other security techniques that slow down transmission are not compliant. As such, IEC 62351-6 does not recommend encryption for IEC 61850 applications. Part 6 of the IEC 62351 standard adds an RSA-based signature to the GOOSE and SV frame to secure the integrity of the frame structure and provide authenticity in installations employing IEC 61850 with ethernet-based technologies. The standard expressly states that the RSASSA-PSS (Probabilistic Signature Scheme) digital signature technique based on RFC 3447 must be used. As RSA signatures are very expensive in terms of the processing power needed, the standard's suggestion to utilise them to secure extended PDUs renders it unsuitable for applications that need a minimum response time of 3 ms. (Ustun et al., 2019) and (Hohlbaum et al., 2010) used the RSA method in accordance with the IEC 62351-6 standard to protect GOOSE messages from cybersecurity threats. However, it has been observed that the computational time for RSA algorithms does not fulfil the 3ms timing criteria, making IEC 61850 message security practically unachievable. (Hohlbaum et al., 2010) further implements the algorithm on a Field Programmable Gate Arrays (FPGA) and Application-Specific Integrated Circuit (ASIC) platform to test performance measures. Both options are inapplicable since the response time is inadequate and the latter is not financially feasible. (Hussain, Ustun, et al., 2020) recommend the implementation of HMAC-SHA256 for securing IEC 61850 protocols which requires less computational time and as such should meet the response time requirements of 3ms. The cybersecurity mechanisms must not cause message delays that exceed the permissible limitations. The most recent iteration of the IEC 62351-6 standard replaced RSA with the Secure Hash Algorithm-256 (SHA-256) and Advanced Encryption Standard (AES) Galois Message Authentication Code (AES-GMAC) algorithms, allowing for both data authentication and encryption. When implemented in hardware, this method has proven to be exceedingly efficient, with high data transmission and minimal latency.

Therefore, implementing IEC 62351 is a continuous process. Once established, security methods must be regularly maintained and upgraded to keep up with changing security risks and challenges to maintain optimal security.

To preserve essential infrastructure and ensure stable power generation, data and communications must be safeguarded. The IEC 62351 standard provides an internationally recognised method for providing that security. To ensure protection in accordance with IEC 62351, smart grids must implement IEC 61850-based systems and devices. IEC 62351 must be used to protect time-critical traffic related to smart grids.

3.6.1 IEC 62351-6: Security Extensions for GOOSE and SV

The "Security for IEC 61850" standard, IEC 62351-6, primarily focuses on the security enhancements for GOOSE and SV.

IEC 62351-6 makes two major contributions to GOOSE and SV security. The first is the inclusion of an optional AES-128 encryption alternative and a new field called *Authentication Value* to GOOSE and SV PDUs that is used to check for integrity. The GOOSE and SV PDUs have been modified to prevent replay assaults as the second security measure. As previously stated, encryption is not encouraged in packets owing to performance considerations. Yet, the usage of encryption permits the prevention of cyber-attacks and data theft. IEC 62351-6 adds two new fields to the GOOSE and SV PDUs:

- *Authentication Value* using RSA digital signature: SHA-256 is used as an input;
- *timestamp* indicates when the PDU was created.

Every PDU must have a valid *Authentication Value* added by the Publisher. As only the genuine publisher is aware of the secret key needed to generate the signature, subscribers can then validate the *Authentication Value* to ensure the authenticity of the PDU. Since both require creating a legitimate signature, attackers are no longer able to manipulate or alter messages as a result. An authentication value enhances the security and reliability of the communication between substations in a power grid. (Robillard, 2018)

3.6.2 Replay Protection for GOOSE

In most cases, a GOOSE message with a greater *stNum* implies that it is the latest. As a result, IEC 62351-6 requires the GOOSE subscriber to record the last *stNum* received from a certain publisher and delete any PDU from this publisher with a lower *stNum*. This approach addresses PDUs that occur out of time, preventing the subscriber from reacting to previous events and perhaps causing problems. Moreover, it defends against replay attacks.

To reset *stNum*, *stNum* is set back to 0 in two cases. In the first case, the 32-bit value overflows to exceed its maximum. The second scenario involves a message timeout. If the subscriber does not get a GOOSE PDU within the time allowed to live (TAL) indicated in the *TAL* field of the most recent PDU received, the subscriber assumes contact has been lost and counts this as a message timeout. *TAL* is a method used to ensure that messages are delivered on time. The message is discarded when the *TAL* value reaches 0.

Time window-based filtering is a popular technique for skew filtering. This entails defining an acceptable time range for GOOSE messages (with a greater *stNum*) to arrive at each node in the network. Messages sent outside of this time frame are deleted. Another option is threshold-based filtering, which requires specifying a skew tolerance as the greatest acceptable deviation from a GOOSE message's predicted arrival time. Because GOOSE messages are used to convey time-sensitive information between IEDs, skew filtering is vital in substation automation systems. If the skew is not managed appropriately, the accuracy and reliability of information interchange might be jeopardised, leading to errors in control and protection operations. Skew filtering considerably restricts replay attacks since PDUs older than the skew period cannot be exploited. (Robillard, 2018)

3.6.3 Replay Protection for SV

A timestamp field is not included in SV PDUs. In most cases, if the publisher and subscriber use synchronised clocks protocol and exchange messages in real-time, this is not an issue. Yet, it is troublesome when we consider the possibility of a delay or replay attack. It is nearly hard to verify the creation time for a certain SV PDU and assess its reliability without the timestamp. To remedy this, IEC 62351-6 includes a timestamp field in every SV packet. This mechanism ensures that the data is not tampered with by verifying the source of the message and the timestamp of the message. The timestamp ensures that the message hasn't been replayed from a previous session.

The *smpCnt* value in SV messages keeps track of the sequence of the PDUs. A PDU with a greater smpCnt is often newer. A technique used to protect against replay attacks is to discard any received SV messages with a *smpCnt* value lower than the previous value, which ensures that only the newest messages are accepted.

One technique to protect against replay attacks in the SV protocol is to reset the *smpCnt* value after a predetermined time interval, when there is a message timeout. Another approach is to detect when the *smpCnt* value overflows and reset the value as needed. By resetting the smpCnt value on every sync pulse, the SV protocol ensures that the sequence of samples is correctly identified, and the correct order of arrival is maintained. This makes it easier to detect and prevent any attempts at replay attacks, which aim to manipulate the sequence of SV messages to create false or misleading data.

Skew filtering for SV PDUs specifies acceptable time tolerances for the arrival of SV messages at each receiver. SV messages arriving outside the specified time range are considered invalid and are discarded. The acceptable time range for SV messages is based on the expected time of arrival, which is typically adjusted to account for delays that can occur due to network congestion or other issues. (Robillard, 2018)

3.7 Conclusion

Applications for protection, automation, and control are the most important aspects of the smart power system's reliability and security. The move from traditional technology to IEC 61850-based solutions in smart substations was made possible by the advancement of communication technology. For new solutions to be employed in existing or new substations, they must be compatible with both traditional and non-traditional technologies (Apostolov, 2020). The provision of implementation instructions ensures interoperability between merging units and primary or secondary equipment. The benefit of employing new technology is that it enables the creation of more advanced communication architectures such as the ethernet-based IEC 61850 communication standard. The suite, in comparison to traditional and hardwired systems, offers several advantages. These benefits make the IEC 61850 standard an excellent choice for communications networks embedded within systems. Consequently, IEC 61850 can be more effective, dependable, and secure than conventional communication techniques in substations, making it a preferable choice.

Time is critical in IEC 61850-based systems, which is why time requirements have been incorporated into object modelling of data, including SV, GOOSE, Client/Server and GSSE.

As mentioned previously, IEC 61850 establishes specifications for event time stamping for use in a variety of protection, automation, and control applications.

In IEC 61850-based protection and control systems, high-speed peer-to-peer topology utilises a mechanism developed to suit a range of specifications. The GSE approach can be thought of as a method for an IED to provide unauthorised reports. The implementation of each IED determines the performance and reliability of the system. (Apostolov, 2020)

Due to the time-critical and high-speed characteristics of today's Ethernet-based communication technology, it is an ideal communication technology for automation applications in substations. Furthermore, an overview of the security threats and vulnerabilities in IEC 61850 GOOSE and SV messages is presented, including the challenges in securing the communication protocols. The security requirements for IEC 61850 messages are also presented. A thorough cyber-physical security solution is necessary in addition to the vulnerability and impact study of cyberattacks to prevent the discovered security issue. A key role is played by the security standard IEC 62351, which is utilised to offer end-to-end Security and secure communication in accordance with IEC 61850 standards.

The knowledge necessary has been generated from the literature reviewed and is believed to be sufficient to enable the proposed research project, which is the development of authentication methods for GOOSE and SV communication.

The following chapter details the project's development and presents the algorithms that will be utilised to complete the project. The implementation procedures developed for client/server functions are explained in detail in Chapter 4.

4. CHAPTER FOUR: CASE STUDY: CONTEXT AND DEVELOPMENT

4.1 Introduction

Security is not a consideration when developing SCADA protocols. These protocols are subject to security threats and, if exploited, could jeopardise the reliability of the power network. When an attacker gains control of a device, they have the ability to change its settings. They can modify the device to the point where the power system becomes unstable and power delivery is disrupted. (Ustun et al., 2020)

Future electrical systems will be represented by active networks capable of bidirectional power transmission. They have improved communication capabilities, allowing them to monitor, protect, and control operations. A phasor measurement unit (PMU) is a device used in power systems to measure and monitor electrical value in real-time. PMUs capture both the magnitude and phase angle of voltage and current, at a specific location in an electrical grid. This means that the measurements are synchronised with a time reference. As such, the term "synchrophasor" emphasizes the synchronization of these measurements across different locations within the electrical grid. The PMU Communication Network (PMU-CN) system is constructed in accordance with the IEEE C37.118.2 and IEC 61850-90-5 standards. IEEE C37.118.2, on the other hand, does not provide any protective mechanisms for mitigating security assaults on an unsecured IP network. Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) protocols used in communication networks between synchrophasors, and Phasor Data Concentrators (PDC) have security vulnerabilities since the protocol that is supposed to be utilised for synchrophasor communication is not defined within the IEEE C37.118.2 standard. As a result, to address the cybersecurity flaws, IEC 61850-90-5 was developed, which includes stronger protection measures and defines HMAC for IEC 61850 message authentication. Effective security solutions must therefore be inclusive, interoperable, and efficient. (Khan et al., 2016)

Based on IEC 61850, Part 6 of the IEC 62351 standard offers security methods to secure real-time communications. SV and GOOSE messages must be generated, transmitted, and analysed in less than 3 ms. After assessing the security risks and challenges to IEC61850 communications and the latest developments in GOOSE and SV cybersecurity, the work presented in Chapter 5, Section 5.5 offers a security algorithm to enable message authentication and confidentiality. The design will be developed and implemented on a Kali-

Linux software environment, where the client-server code and security algorithm for R-SV will be evaluated. As a result, extensive knowledge is necessary for raw socket programming, client-server code design, and GOOSE and R-SV message structure. Objectives of the design phase seek to adapt and improvise an existing security algorithm as presented by (Hussain, Farooq, et al., 2020) for IEC 61850 messages and investigate the performance of the algorithm. The article investigates the security issues with the IEC 61850 standard's GOOSE messages and suggests a technique to ensure the messages' secrecy and integrity. The article then discusses the potential flaws in GOOSE communications and the necessity of incorporating security. To achieve confidentiality and integrity in the GOOSE messages, the approach suggested employs an arrangement of symmetric encryption and digital signatures. The article contains the method's technical details as well as a full study of the proposed method's security features. The study also compares the proposed method to other current methods and highlights the benefits of the proposed method in terms of security, performance, and compatibility with existing systems. In its entirety, the article offers a thorough analysis of the security issues relating to the GOOSE messages in the IEC 61850 standard and suggests a workable strategy to ensure the confidentiality and integrity of these messages. The suggested approach can make the electrical power system significantly more secure and guarantee the power grid's dependable and secure operation.

In this chapter, we discuss socket programming and further study the toolbox called Routable Goose/Sampled Value (R-GoSV) and Secure Goose/Sampled Value (S-GoSV) as presented by (Ustun et al., 2020) and (Farooq; et al., 2019). Section 4.2 mainly focuses on the development and implementation of the client/server communication component of the design architecture. Section 4.3 discusses GOOSE message structures according to IEC 61850-8-1 and shows implementation details of plain GOOSE software. Section 4.4 describes the implementation details of secure GOOSE communication, and further demonstrates the results with Wireshark capture.

4.2 TCP Client-Server Socket Programming

Sockets act as virtual endpoints for all network communications between two hosts connected by a network. TCP interfaces are connection-oriented, which means they support the idea of an isolated connection on a specific port that can only be used by one program at a time. Because of the connection concept, TCP is a reliable stream; if errors occur, they may be identified and compensated for by resending the rejected packets. (Moon & posts by Silver Moon & rarr.; 2020)

TCP client-server programming is a networking technique used for implementing communication using the IEC 61850 protocol in electrical substations. The client-server programming model involves the use of a client application and a server application, with the client sending requests to the server and the server responding with the requested information.

One of the key strengths of TCP client-server programming for IEC 61850 communication is its compatibility with a wide range of devices and systems. TCP is a widely used and well-established protocol, making it a reliable option for communication in different types of substations. Additionally, TCP is a reliable protocol that ensures data is transferred accurately and efficiently, making it a good choice for critical systems.

Another strength of TCP client-server programming is its flexibility. The client and server applications can be customised and configured to meet specific requirements, providing more flexibility for users. Additionally, TCP client-server programming can be implemented across different types of networks, making it a versatile option for communication in different types of environments.

However, there are also some limitations to TCP client-server programming for IEC 61850 communication. One major limitation is its lack of security features. TCP communication is vulnerable to cyber threats and attacks, which can be particularly problematic in the energy sector where the impact of such attacks can be significant. Therefore, additional security measures may need to be implemented to ensure secure communication.

Another limitation of TCP client-server programming is its reliance on a stable network connection. If the network connection is lost or unstable, communication can be disrupted, leading to errors and even system failures. The design and implementation of TCP client-server programming for IEC 61850 communication involves several key considerations:

- Firstly, the design of the TCP client-server architecture should take into account the specific requirements of the IEC 61850 standard. This includes defining the data model, data exchange mechanisms, and communication profiles required for the system.
- Secondly, the implementation of the TCP client-server protocol should ensure the reliability and security of the communication. This includes implementing error checking and correction codes, as well as encryption and authentication to protect against attacks such as eavesdropping and tampering.

- Thirdly, the implementation should consider the performance and scalability of the system. This includes optimising the network bandwidth, minimising latency, and ensuring that the system can handle large amounts of data and multiple concurrent connections.

4.2.1 Common Functions used in Socket Programming

The ***socket()*** function creates a client socket, which is then connected to a remote address with the ***connect()*** function, and lastly data can be retrieved with the ***recv()*** function. On the server end, we must also establish a socket with a ***socket()*** call, but we must then bind() that socket to an IP and port where it may ***listen()*** for connections, ***accept()*** connections, and then ***send()*** or ***recv()*** data to the other interfaces on the network. (Moon & posts by Silver Moon & rarr.; 2020)

4.2.2 Implementation of Raw Socket

Raw socket programming is a low-level networking technique used for implementing communication using the IEC 61850 protocol in electrical substations. Raw socket programming allows the developer to directly access the network interface, bypassing the operating system's network stack and providing more control over the communication process. The strengths and limitations of raw socket programming are discussed below.

Strengths:

1. One of the key strengths of raw socket programming for IEC 61850 communication is its flexibility and customisation. Raw socket programming allows the developer to customise the communication process and control the data transmission and reception, providing more flexibility for users. Additionally, raw socket programming can be used with different types of network interfaces, making it a versatile option for communication in different types of environments.

2. Another strength of raw socket programming is its efficiency. Raw socket programming can achieve high performance and low latency because it allows the developer to directly access the network interface, bypassing the operating system's network stack, which can introduce additional processing delays.

Limitations:

1. However, there are also some limitations to raw socket programming for IEC 61850 communication. One major limitation is its complexity. Raw socket programming requires a high level of expertise and understanding of network protocols and systems, which can be a barrier for some users. Additionally, raw socket programming can be more prone to errors and vulnerabilities, as it bypasses some of the built-in security features of the operating system's network stack.

2. Another limitation of raw socket programming is its lack of compatibility with some systems and devices. Raw socket programming may not be compatible with some legacy systems or devices that require higher-level networking protocols.

Socket interfaces are those that circumvent the TCP/IP layers and instead transmit packets to the designated application via the Internet Control Message Protocol (ICMP) and Internet Group Management Protocol (IGMP). This enables the program to build ICMP and IGMP purely as user processes rather than injecting additional code into the kernel.

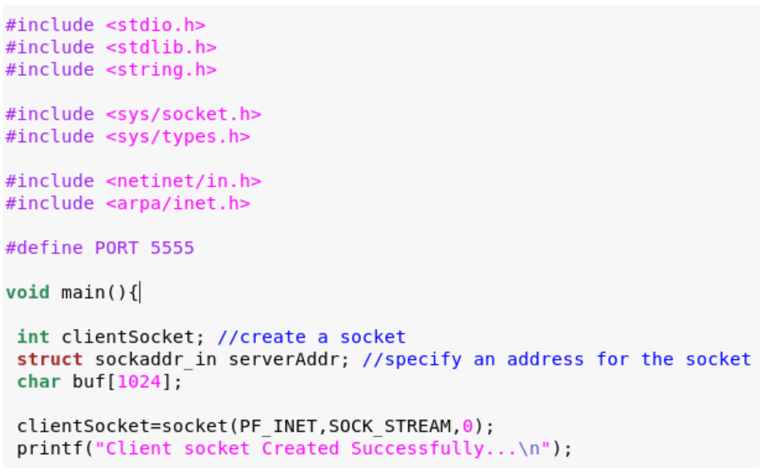
4.2.3 Socket Programming – Case Study

Below you'll find a case study of a very simple client-server software in C. The client establishes a link to the server, and the server delivers the message "Hello From Me", and the client outputs the received message. These programs are often labelled "Client" and "Server". The "Client" shares the information with the "Server" by sending its messages through an Ethernet port.

The source code found in Appendix A and Appendix B is used to analyse the client and server programs to open a socket. The socket must be configured by both the client and server. Socket family, socket type, and protocol are all necessary.

1. *int socket(int domain, int type, int protocol);*

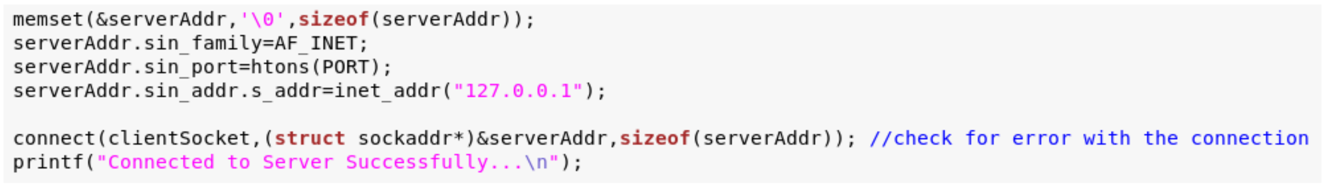
Refer to Figure 4.1: The socket family for a raw socket is ***PF_INET***, the socket type for TCP is ***SOCK_STREAM*** and for the protocol, the ***netinet/in.h*** defines the ***sockaddr_in*** structure. The ***sockaddr_in*** structure is used to store addresses for the Internet address family. (Saxena, 2015)



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <sys/socket.h>
#include <sys/types.h>

#include <netinet/in.h>
#include <arpa/inet.h>

#define PORT 5555

void main(){

 int clientSocket; //create a socket
 struct sockaddr_in serverAddr; //specify an address for the socket
 char buf[1024];

 clientSocket=socket(PF_INET,SOCK_STREAM,0);
 printf("Client socket Created Successfully...\n");
```

Figure 4.1: Client socket

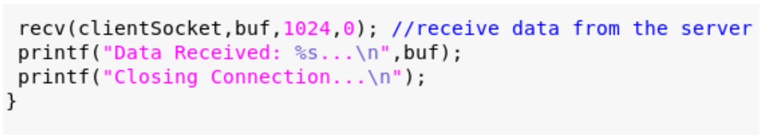
Refer to Figure 4.2: If the socket creation succeeds, it will return "Client socket Successfully Created." After the socket has been allocated, it must connect to the server. A ***sockaddr_in*** structure describing the server is required for the connection to be formed. Specifically, we must use ***serverAddr.sin*** and ***serverAddr.sin*** port to indicate the server and port to connect to. The IP address is supplied via ***serverAddr.sin*** family, configured to ***AF_INET***.



```c
memset(&serverAddr,'\0',sizeof(serverAddr));
serverAddr.sin_family=AF_INET;
serverAddr.sin_port=htons(PORT);
serverAddr.sin_addr.s_addr=inet_addr("127.0.0.1");

connect(clientSocket,(struct sockaddr*)&serverAddr,sizeof(serverAddr)); //check for error with the connection
printf("Connected to Server Successfully...\n");
```

Figure 4.2: Connect function

Refer to Figure 4.3: As with creating the socket, an established connection will return “Connected to Server Successfully”. Data can now be sent and received through the socket.

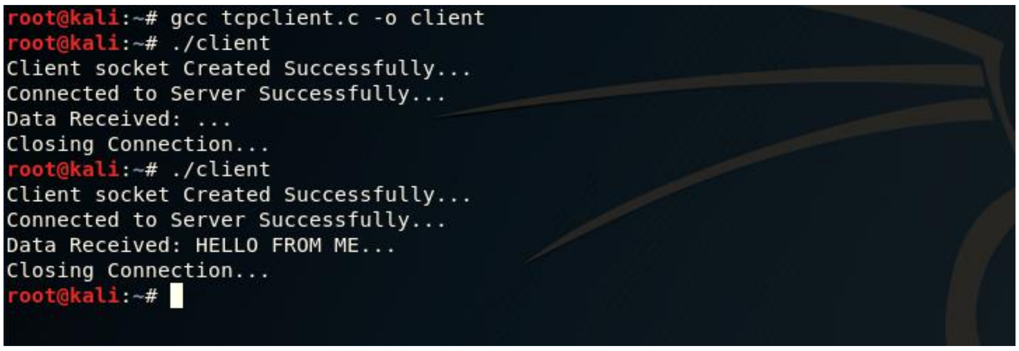


```c
 recv(clientSocket,buf,1024,0); //receive data from the server
 printf("Data Received: %s...\n",buf);
 printf("Closing Connection...\n");
}
```

Figure 4.3: Close function

The receive function must receive data from a connection-mode or connectionless-mode socket. Because it prevents the program from obtaining the source address of incoming data, it is most typically used with linked sockets.

At the terminal, compile and run the program. To run and compile the software, one requires administration rights as a root user. The client socket's output is shown in Figure 4.4. Client sockets were successfully generated and connected.

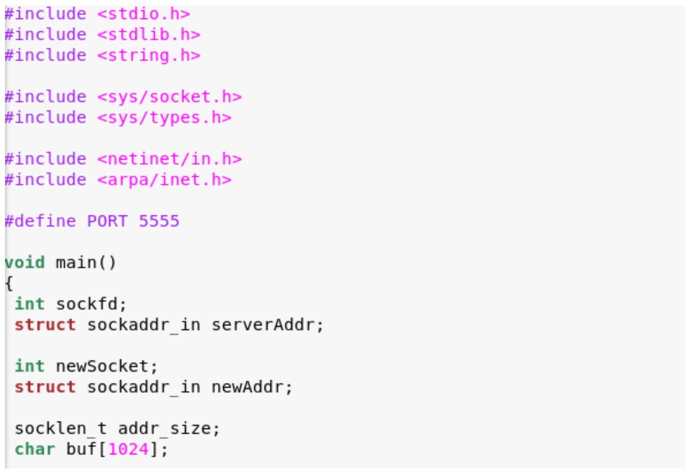


```
root@kali:~# gcc tcpclient.c -o client
root@kali:~# ./client
Client socket Created Successfully...
Connected to Server Successfully...
Data Received: ...
Closing Connection...
root@kali:~# ./client
Client socket Created Successfully...
Connected to Server Successfully...
Data Received: HELLO FROM ME...
Closing Connection...
root@kali:~# 
```

Figure 4.4: Software output for Client communication

As previously stated, the goal of generating a socket for a server differs from that of a client. The socket is created using the same syntax as the client, except that the structure is set up with information pertaining to the server rather than the peer it wants to connect to.

Usually, the special contact ***INADDR_ANY*** may be used to enable receiving client requests on any IP address the server supplies; in principle, such as in a multi-hosting server, you could specify a particular IP address as illustrated in Figure 4.5.

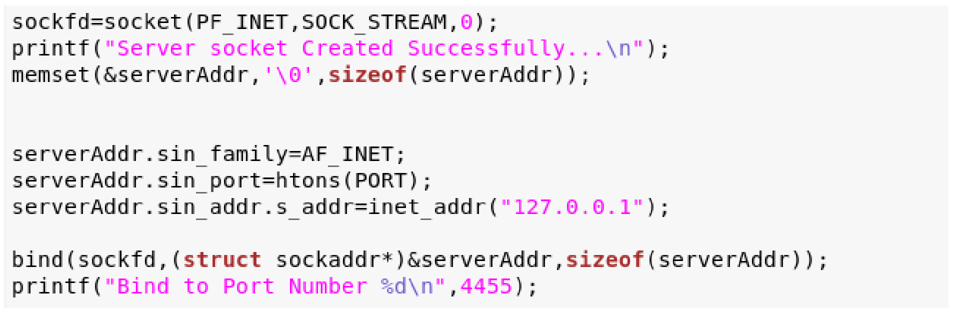


```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <sys/socket.h>
#include <sys/types.h>

#include <netinet/in.h>
#include <arpa/inet.h>

#define PORT 5555

void main()
{
 int sockfd;
 struct sockaddr_in serverAddr;

 int newSocket;
 struct sockaddr_in newAddr;

 socklen_t addr_size;
 char buf[1024];
```

Figure 4.5: Server socket

A server must be able to manage several client requests, it is more sophisticated than a client. A server essentially has two functions: managing existing connections and listening for new ones to form.

As illustrated in Figure 4.6, the server binds to its address and port, while the client program connects to the server's IP address and port. It is now possible to listen to the server socket after it has been bound. Once a server socket is listening, it is able to receive client connections.

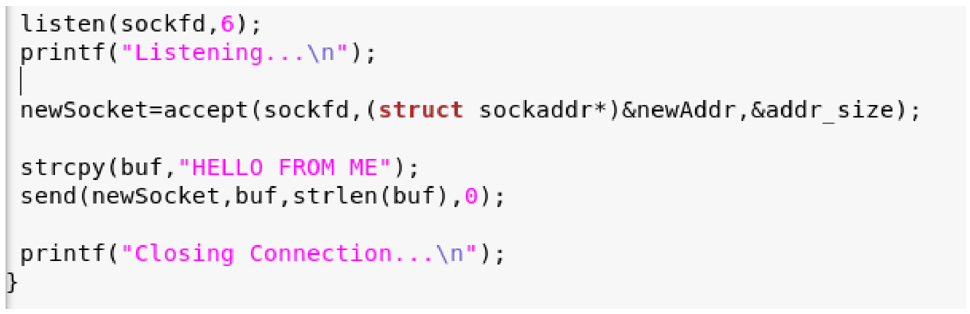


```
sockfd=socket(PF_INET,SOCK_STREAM,0);
printf("Server socket Created Successfully...\n");
memset(&serverAddr,'\0',sizeof(serverAddr));


serverAddr.sin_family=AF_INET;
serverAddr.sin_port=htons(PORT);
serverAddr.sin_addr.s_addr=inet_addr("127.0.0.1");

bind(sockfd,(struct sockaddr*)&serverAddr,sizeof(serverAddr));
printf("Bind to Port Number %d\n",4455);
```

Figure 4.6: Receive function

A bind function obtains a unique name for the socket when the socket descriptor is generated. The listen function enables the server to accept connections from clients. The backlog is fixed at six in this case study. This implies the system will queue up to six incoming connection requests before rejecting them as shown in Figure 4.7.

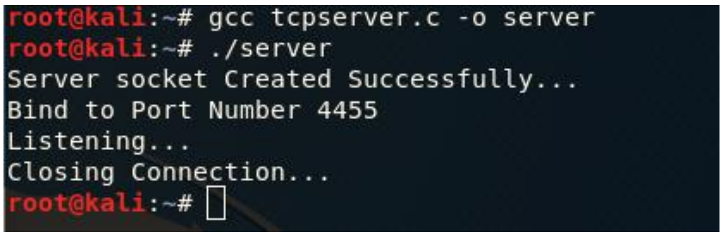
To accept an incoming connection request, the server utilises the accept function. The accept call will remain blocked indefinitely while the incoming connection is established.



```
 listen(sockfd,6);
 printf("Listening...\n");

 newSocket=accept(sockfd,(struct sockaddr*)&newAddr,&addr_size);

 strcpy(buf,"HELLO FROM ME");
 send(newSocket,buf,strlen(buf),0);

 printf("Closing Connection...\n");
}
```

Figure 4.7: Bind function

At the terminal, compile and execute the program as depicted in Figure 4.8. Keep in mind that you should execute the software as root. To use raw sockets, you'll need to be logged in as root. The socket was successfully created, as evidenced by the output of the server socket.



```
root@kali:~# gcc tcpserver.c -o server
root@kali:~# ./server
Server socket Created Successfully...
Bind to Port Number 4455
Listening...
Closing Connection...
root@kali:~# 
```

Figure 4.8: Software output for Server communication

In conclusion, the successful production and transmission of packets over a network must be confirmed by using packet sniffers like Wireshark. To comprehend the data flow through network connections, it is also crucial to understand the roles of clients and servers.
We can discover a lot about how clients and servers communicate by examining the network traffic that a packet sniffer has gathered. We may monitor the exchange of data packets, monitor network protocols, and identify potential problems or anomalies that may occur during transmission.

In our case study, we looked at the functions of a client and a server concerning web browsing. We learned that the server is a remote computer that a client—like a web browser—creates a connection. Through sockets, the server accepts connections from clients and responds to them, allowing the client to request data retrieval.

In the specific example we explored, the web browser acted as the client, and www.google.com served as the server. This relationship demonstrated how clients and servers work together to enable seamless data exchange and provide users with the information they seek.

Network administrators and developers may efficiently debug network issues, optimise performance, and maintain the smooth operation of diverse applications and services by using tools like Wireshark and comprehending the client-server architecture. In conclusion, the analysis of network traffic using packet sniffers and the understanding of clients and servers play vital roles in the field of networking. We can increase network communications' effectiveness and dependability by further study and investigation of these ideas, which will result in better user experiences and more durable digital infrastructure.

The IEC 61850 protocol has specific communication and behaviour requirements, and the communication middleware architecture is created to meet those requirements. A thorough examination of the model, including its architectural components, was presented in Chapter 3.2. The design only supports a limited number of communication techniques required for the successful operation of ACSI clients and servers. The case study above showed that the developed architecture may be used successfully to offer the communication services required by the IEC 61850 protocol.

4.3 Plain GOOSE Source Code

Appendix C shows the plain GOOSE source code, which creates R-GOOSE packets without applying any security algorithms to them and sends them to the network with their entire stack of headers from IP, UDP, and session layers, followed by application data. The application data in this case is a GOOSE protocol as illustrated in IEC 61850-8-1.

Intelligent electronic devices (IEDs) that use Ethernet communication protocols are becoming standard in electric substations. One of the most crucial aspects of power system operation is the communications infrastructure that supports power system protection, monitoring, and control. With the emergence of IEDs, there is a growing requirement for communication to be provided to these devices. Communication without security techniques presents several network concerns. A negative consequence of this is exposure to cyberattacks. Ethernet-based networks are easier to gain access to, and standardised communications allow hackers to know exactly what commands to provide. The following is a list of strengths and weaknesses of non-secure GOOSE.

Strengths:

- Fast and efficient: GOOSE messages are multicast, which means that they can be sent to multiple devices simultaneously, reducing the amount of traffic on the network and improving the response time.
- Reliable: GOOSE messages include error checking and correction codes, ensuring the integrity of the information transmitted.
- Scalable: GOOSE messages can be used to exchange different types of information, such as status, commands, and measurements, making it a flexible protocol that can be adapted to different applications.
- Interoperable: The IEC 61850 standard defines the format and structure of GOOSE messages, ensuring interoperability between devices from different vendors.

Limitations:

- Security: GOOSE messages are not encrypted, making them vulnerable to eavesdropping, tampering, and replay attacks. Additional security measures, such as encryption and authentication, must be implemented to ensure the confidentiality and integrity of the information transmitted.

- Network bandwidth: Although GOOSE messages are efficient, they still require bandwidth on the network, and excessive GOOSE traffic can congest the network and cause delays.
- Configuration: The configuration of GOOSE messages can be complex, requiring specialised knowledge and tools to set up and maintain.

Overall, the GOOSE source code algorithm is a reliable and flexible protocol for communication in power substations. However, its limitations in terms of security and complexity must be addressed when implementing it in a substation. Additional security measures and network management strategies should be used to ensure the secure and efficient operation of the protocol. This can be a significant concern in critical power system applications where data security is a top priority.

To accomplish interoperability and standardisation, the message format or structure must be specified. To skip the network and transport layer headers, GOOSE communication is directly mapped to the data link layer, reducing the size of the message and, as a result, the propagation and processing delays of the GOOSE messages. The GOOSE message, as illustrated in Figure 4.9, consists of six-byte destination and source address fields, followed by a two-byte Ether-type field that specifies the type of data in the payload field. The ether-type value for a GOOSE message is 88-B8. APPID, Length, Reserved1, Reserved2, and GOOSE APDU fields are followed by padded data and Frame Check Sequence in the GOOSE PDU (FCS). The source code supplied in Appendix C can be used to verify the GOOSE PDU format illustrated in Figure 4.9.

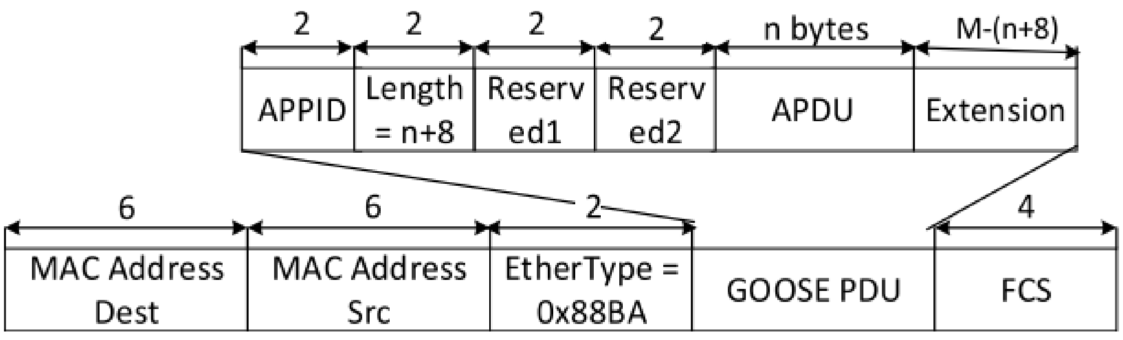


Figure 4.9: GOOSE PDU format (Hussain et al., 2019)

Figure 4.11 shows the Wireshark capture of the unsecure GOOSE generated. It shows all the relative fields of the GOOSE message format. The TCP Ethernet communication programming was presented above in Section 4.2 to illustrate that all the communication traffic is sent and received through the Ethernet port. The simple TCP sending and receiving example presented above is used as a reference to understand how to send out information through the Ethernet port.

The plain GOOSE source code presented by (Hussain et al., 2019) uses the UDP. UDP is a transport layer protocol with enables application programs to send messages to each other. The UDP protocol also has two applications, namely the client and the server. The client sends out UDP messages to the server through the Ethernet port. Since the aim of the program is to send out GOOSE messages, the concentration is put on the client (sending) program. The UDP sending is illustrated on a flow chart in Figure 4.10.

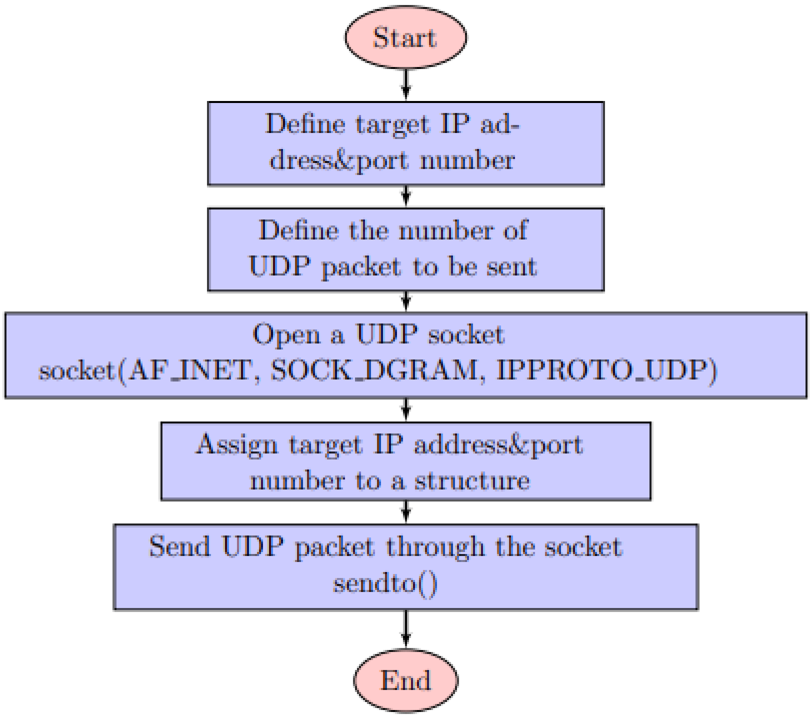
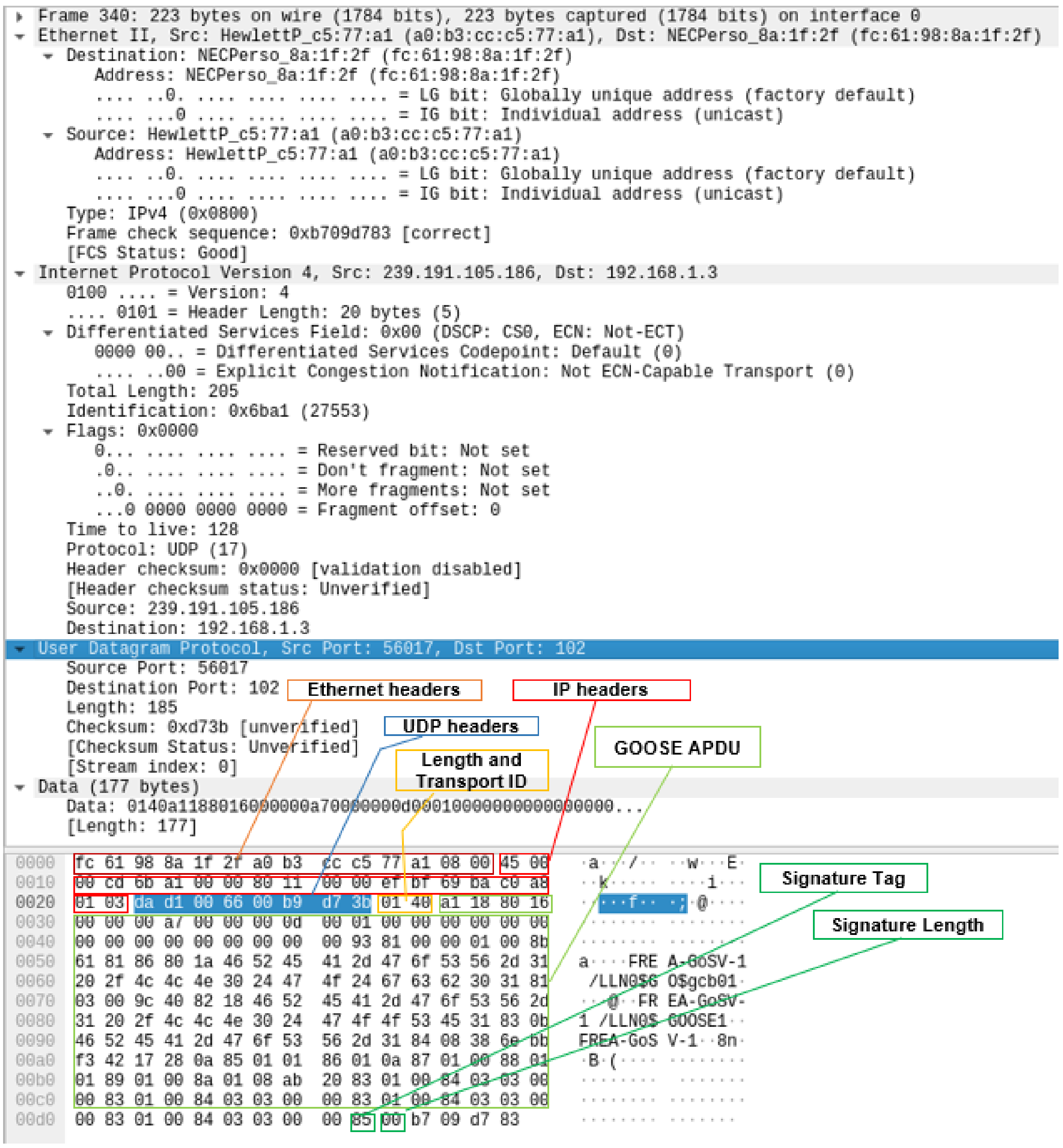

Start
Define target IP address&port number
Define the number of UDP packet to be sent
Open a UDP socket
socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)
Assign target IP address&port number to a structure
Send UDP packet through the socket
sendto()
End

Figure 4.10: UDP Sending (Zhao, 2012)



```
▸ Frame 340: 223 bytes on wire (1784 bits), 223 bytes captured (1784 bits) on interface 0
▾ Ethernet II, Src: HewlettP_c5:77:a1 (a0:b3:cc:c5:77:a1), Dst: NECPerso_8a:1f:2f (fc:61:98:8a:1f:2f)
  ▾ Destination: NECPerso_8a:1f:2f (fc:61:98:8a:1f:2f)
      Address: NECPerso_8a:1f:2f (fc:61:98:8a:1f:2f)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  ▾ Source: HewlettP_c5:77:a1 (a0:b3:cc:c5:77:a1)
      Address: HewlettP_c5:77:a1 (a0:b3:cc:c5:77:a1)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
    Frame check sequence: 0xb709d783 [correct]
    [FCS Status: Good]
▾ Internet Protocol Version 4, Src: 239.191.105.186, Dst: 192.168.1.3
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▾ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      0000 00.. = Differentiated Services Codepoint: Default (0)
      .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    Total Length: 205
    Identification: 0x6ba1 (27553)
  ▾ Flags: 0x0000
      0... .... .... .... = Reserved bit: Not set
      .0.. .... .... .... = Don't fragment: Not set
      ..0. .... .... .... = More fragments: Not set
      ...0 0000 0000 0000 = Fragment offset: 0
    Time to live: 128
    Protocol: UDP (17)
    Header checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source: 239.191.105.186
    Destination: 192.168.1.3
▾ User Datagram Protocol, Src Port: 56017, Dst Port: 102
    Source Port: 56017
    Destination Port: 102
    Length: 185
    Checksum: 0xd73b [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
▾ Data (177 bytes)
    Data: 0140a1188016000000a7000000d0001000000000000000000...
    [Length: 177]

0000  fc 61 98 8a 1f 2f a0 b3  cc c5 77 a1 08 00 45 00   ·a··/··· ··w···E·
0010  00 cd 6b a1 00 00 80 11  00 00 ef bf 69 ba c0 a8   ··k····· ····i···
0020  01 03 da d1 00 66 00 b9  d7 3b 01 40 a1 18 80 16   ·····f·· ·;·@····
0030  00 00 00 a7 00 00 00 0d  00 01 00 00 00 00 00 00   ········ ········
0040  00 00 00 00 00 00 00 00  00 93 81 00 00 01 00 8b   ········ ········
0050  61 81 86 80 1a 46 52 45  41 2d 47 6f 53 56 2d 31   a····FRE A-GoSV-1
0060  20 2f 4c 4c 4e 30 24 47  4f 24 67 63 62 30 31 81    /LLN0$G O$gcb01·
0070  03 00 9c 40 82 18 46 52  45 41 2d 47 6f 53 56 2d   ···@··FR EA-GoSV-
0080  31 20 2f 4c 4c 4e 30 24  47 4f 4f 53 45 31 83 0b   1 /LLN0$ GOOSE1··
0090  46 52 45 41 2d 47 6f 53  56 2d 31 84 08 38 6e bb   FREA-GoS V-1··8n·
00a0  f3 42 17 28 0a 85 01 01  86 01 0a 87 01 00 88 01   ·B·(···· ········
00b0  01 89 01 00 8a 01 08 ab  20 83 01 00 84 03 03 00   ········ ········
00c0  00 83 01 00 84 03 03 00  00 83 01 00 84 03 03 00   ········ ········
00d0  00 83 01 00 84 03 03 00  00 85 00 b7 09 d7 83      ········ ·······
```

Figure 4.11: Packet capture of unsecure GOOSE message

Two crucial functions, ***socket()*** and ***sendto()*** are used to send UDP messages via the Ethernet port, as shown in the source code forward in Appendix C.

The program uses the socket function to transmit and receive packets, as well as perform other socket actions as shown in Figure 4.12. The socket function opens a socket communication and returns a descriptor when it is invoked. On success, this method returns the file descriptor; on failure, it returns -1. This function's call format is as follows:

1. *int socket(int domain, int type, int protocol);*

The domain value, as stated with the TCP protocol, provides the protocol family that will be used for communication. This option is set to ***AF_INET*** in this implementation, indicating that the address family is Internet Protocol version 4 (IPv4). The function's type is indicated, and it's utilised to establish the communication semantics. In this implementation, it's set to ***SOCK_RAW***, which enables datagrams. The protocol argument is the third parameter, and it provides the protocol that will be used with the function. This option is ***IPPROTO_RAW*** for the UDP protocol.

```
int main(int argc, char *argv[])
{
    int sfd;
    int i=0,j=0;
    struct ifreq if_idx;
    struct ifreq if_mac;
    int tx_len;
    unsigned char sendbuf[B_SIZE];
    struct sockaddr_ll socket_address;
    char ifName[IFNAMSIZ];

    /* Get interface name */
    strcpy(ifName, IF_NAME);

    /* Open RAW socket to send on */
    if ((sfd = socket(AF_PACKET, SOCK_RAW, IPPROTO_RAW)) == -1)
    {
        perror("socket");
    }

    /* clear the struct ifreq if_idx with memset system call */
    memset(&if_idx, 0, sizeof(struct ifreq));

    /* copy interface name into struct ifreq if_idx */
    strncpy(if_idx.ifr_name, ifName, IFNAMSIZ-1);

    /* configure the interface index */
    if (ioctl(sfd, SIOCGIFINDEX, &if_idx) < 0)
        perror("SIOCGIFINDEX");
```

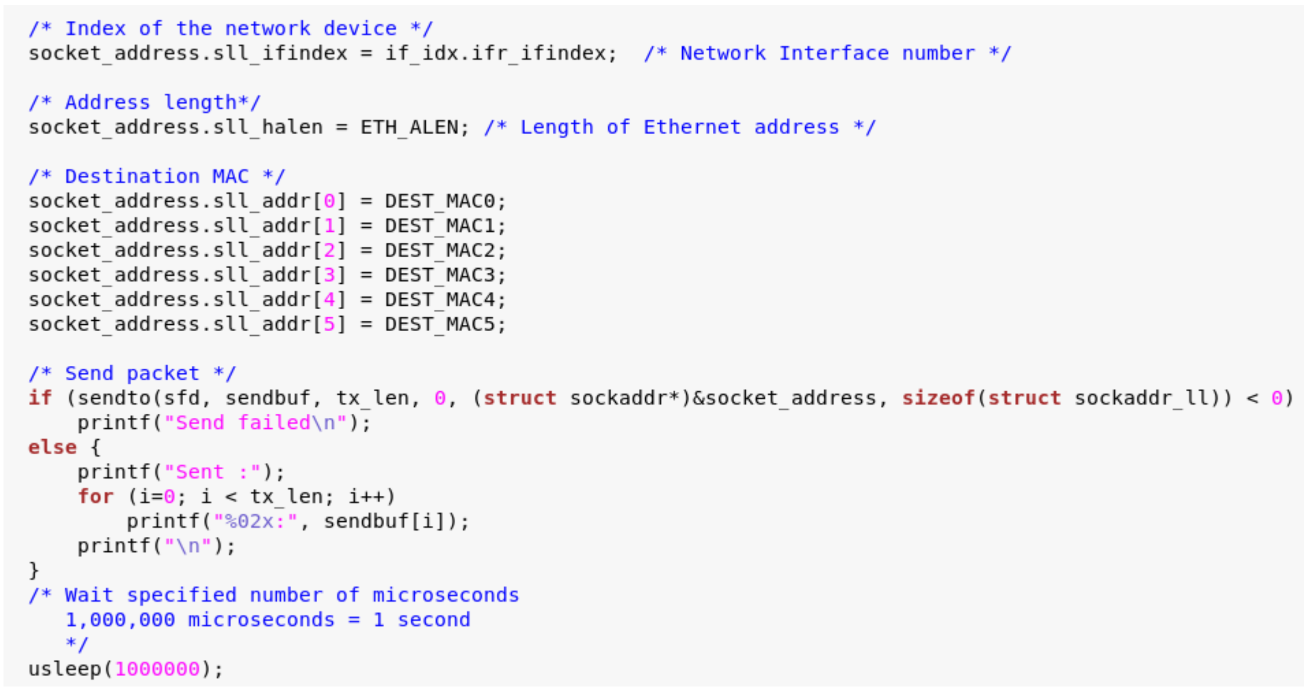
Figure 4.12: Create socket function

The send function is utilised to send data to a different socket. The format for calling this function is as follows:

2. *ssize_t sendto(int s, const void *buf, size_t len, int flags, const struct sockaddr *to_addrs, socklen_t to_len);*

Refer to Figure 4.13, the option ***s*** is the sending socket's file descriptor. This message can be located in ***buf*** and has a length of ***len*** characters. In the case of UDP, the flags parameter is the bit string OR of zero, and it is used to identify the message transmission type. The ***to_addrs*** option links to the ***sockaddr*** structure, which stores the destination address. The

to_len option specifies the length of the target ***sockaddr***. If the function is successful, it will return the number of characters sent; otherwise, it will return -1.



```c
/* Index of the network device */
socket_address.sll_ifindex = if_idx.ifr_ifindex;  /* Network Interface number */

/* Address length*/
socket_address.sll_halen = ETH_ALEN; /* Length of Ethernet address */

/* Destination MAC */
socket_address.sll_addr[0] = DEST_MAC0;
socket_address.sll_addr[1] = DEST_MAC1;
socket_address.sll_addr[2] = DEST_MAC2;
socket_address.sll_addr[3] = DEST_MAC3;
socket_address.sll_addr[4] = DEST_MAC4;
socket_address.sll_addr[5] = DEST_MAC5;

/* Send packet */
if (sendto(sfd, sendbuf, tx_len, 0, (struct sockaddr*)&socket_address, sizeof(struct sockaddr_ll)) < 0)
    printf("Send failed\n");
else {
    printf("Sent :");
    for (i=0; i < tx_len; i++)
        printf("%02x:", sendbuf[i]);
    printf("\n");
}
/* Wait specified number of microseconds
   1,000,000 microseconds = 1 second
   */
usleep(1000000);
```

Figure 4.13: Send function

UDP communications can be transmitted across the Ethernet port using these two basic functionalities. The software Wireshark can capture Ethernet traffic. Figure 4.10 depicts the capturing of UDP packets. It is essential to notice that data such as the Ethernet header and the protocol tag are also included in the whole UDP packet containing the message to be conveyed. This data encapsulation can be done automatically for UDP because it is a transportation layer protocol, but it cannot be done for the data link layer. To comprehend how to transfer data via the Ethernet port, the UDP sending and receiving example is utilised as a guide.

4.4 R-GoSV GOOSE Source Code

This C program, as demonstrated in Appendix D, combines IP, UDP, Session layer headers, and application data to send a full stack of GOOSE APDU messages created according to IEC 61850-90-5. The GOOSE message is classified as application data by IEC 61850-8-1. The session layer includes the needed security fields. Information from the session layer is encrypted and transferred over the network, along with a digital signature.

This toolbox was developed by (Ustun et al., 2020) utilising the SSL programming library that secures R-GOOSE and R-SV communications that can be transmitted in a network without any security mechanism. The proposed R-GoSV algorithms can be used to generate R-GOOSE and R-SV packets with security which can then be used for testing various cybersecurity techniques.

Furthermore, since the communication traffic is sent and received via the Ethernet port, the data link layer programming is studied for R-SV communication. To send data link layer traffic out, all relevant data must be programmed before the calling of sending functions. An Address Resolution Protocol (ARP) sample code is examined to understand the programming of the data link layer packet (Zhao, 2012). An ARP socket is sent via the socket function and a buffer argument is assigned to the sendto function. It is necessary to modify the socket command to allow the sendto function to send out ARP structures. The Ethernet port will be utilised to transmit data. The R-SV protocol implementation will be an extension of the ARP sending program. However, the ARP structure must be modified to conform to the IEC 61850-9-2 standard.

GoSV framework has been developed to build GOOSE and SV messages from scratch. Manually building these messages with low-level detail enables cutting-edge research where novel approaches are tested (Ustun, 2021). Routable messages such as R-GOOSE and R-SV are an addition as per IEC 61850-90-5. These messages are used to secure PMU communication and follow the structure shown in Figure 4.14.

The R-GOOSE source code algorithm uses the standard GOOSE message format and adds several additional fields to support routing and other features. The algorithm is designed to be simple and efficient, with a low processing overhead and minimal impact on network performance. One of the main strengths of the R-GOOSE source code algorithm is its flexibility. The algorithm can be implemented in a wide range of network environments, including both Ethernet and IP-based networks. This makes it well-suited for use in a variety of different power system applications, from small distribution networks to large-scale transmission systems. Another strength of the R-GOOSE source code algorithm is its security features. The algorithm includes support for authentication and encryption, which helps to prevent unauthorised access to GOOSE messages and protect the integrity of the data. This makes R-GOOSE a suitable choice for critical power system applications (Wide-Area Monitoring Systems, Distributed Energy Resources, Substation Automation systems) where data security is paramount.

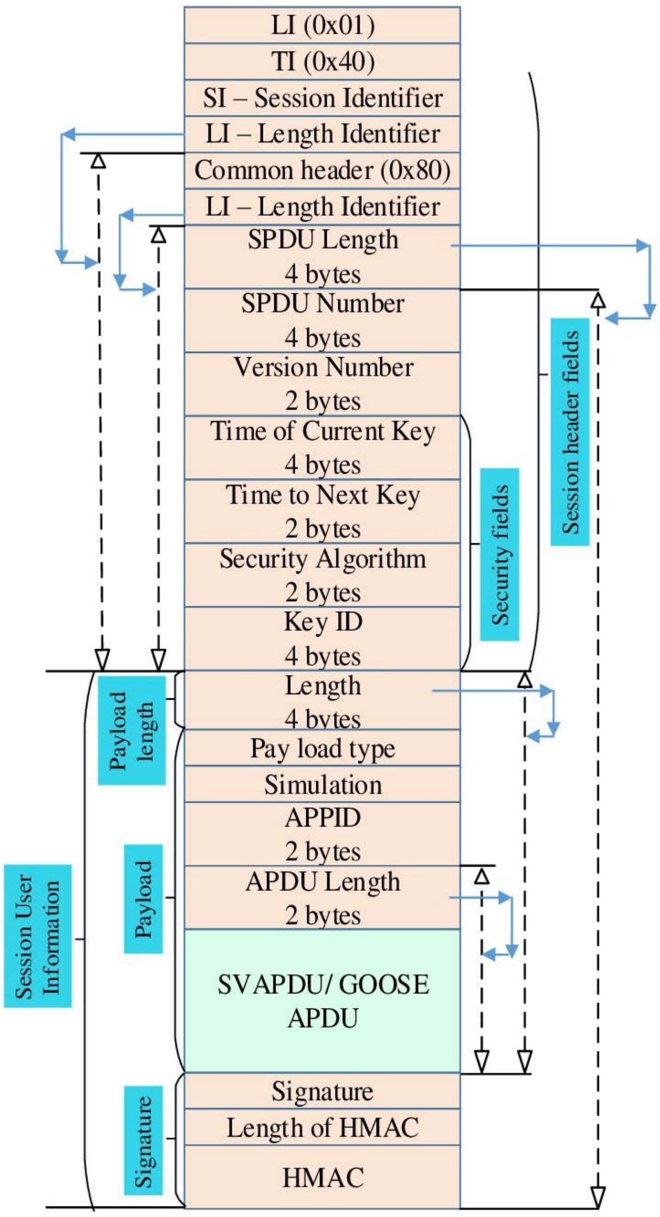


Figure 4.14: Session Layers in IEC 61850-90-5 (Ustun, 2021)

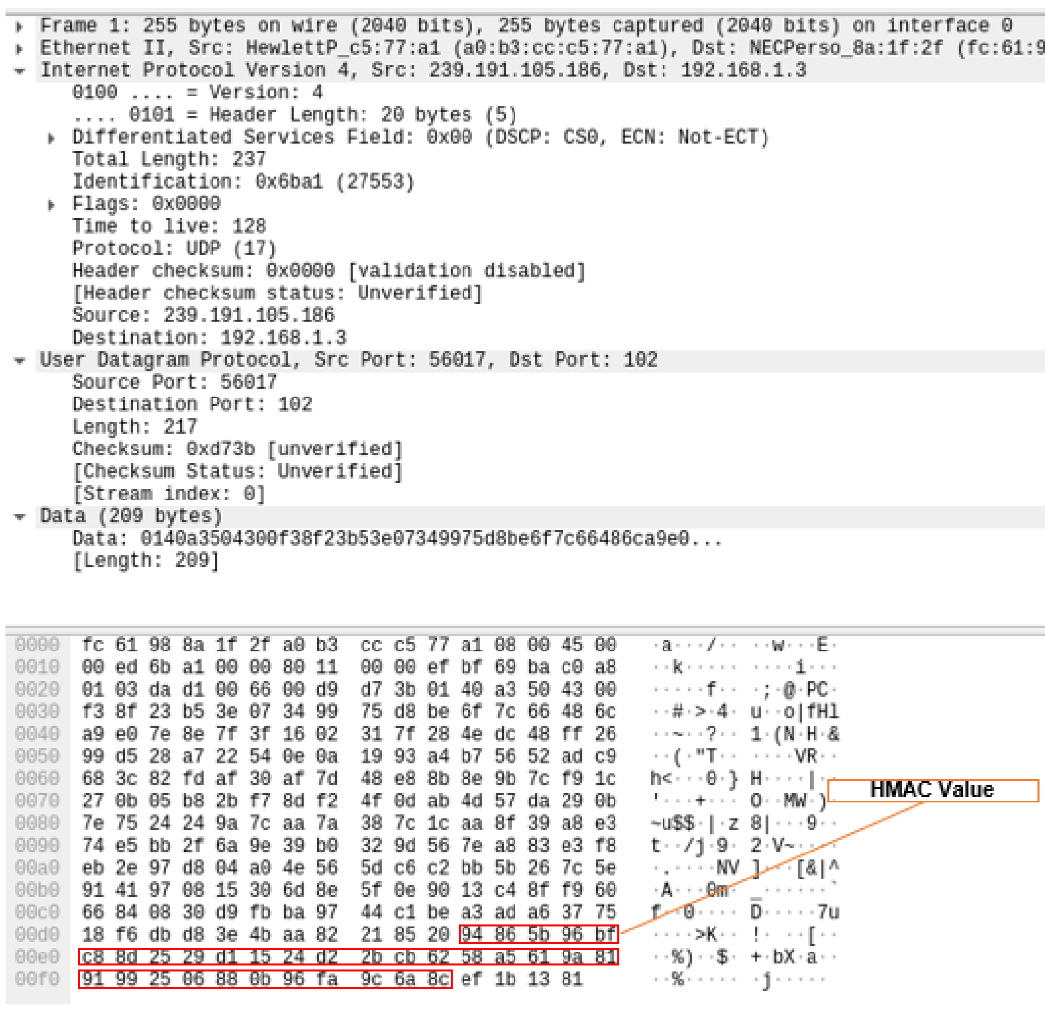
Once implemented, the R-GoSV toolbox is able to send regular and secured (encrypted) R-GOOSE/R-SV messages as shown in Figure 4.15, respectively. It is important to highlight here that, in the latter case, the message cannot be successfully detected since the contents are encrypted. The IEC 61850-90-5 standard describes MAC techniques for generating hash values to achieve message integrity and authentication. Despite confidentiality not being a requirement, the standard specifies encryption algorithms for IEC 61850-90-5 R-GOOSE and R-SV messages. As depicted in Figure 4.15, the Wireshark capture shows all the required fields starting from ethernet, IP, UDP and Session headers. Furthermore, the strengths and weaknesses of the R-GoSV messages are discussed below:

Strengths:

- Security: R-GoSV messages are encrypted and authenticated, which makes them more secure than standard GOOSE messages. The use of digital signatures ensures that the information is authentic and has not been tampered with. It provides a secure method of communication that protects against cyber threats and attacks, which is crucial in the energy sector where the impact of such attacks can be significant. Additionally, the R-GoSV algorithm provides validation of messages to ensure that only valid messages are accepted, further enhancing the security of the communication.
- Flexibility: R-GoSV messages can be routed over IP networks, making it possible to communicate between substations and different parts of the power grid. This feature can be particularly useful in larger substations where longer distances need to be covered.
- Interoperability: The R-GoSV source code algorithm is defined by the IEC 61850 standard, ensuring interoperability between devices from different vendors.

Limitations:

- Complexity: The configuration of R-GoSV messages can be complex, requiring specialised knowledge and tools to set up and maintain. This can make it challenging to implement and troubleshoot. If not implemented correctly, it can lead to communication errors and even system failures.
- Overhead: R-GoSV messages require additional processing overhead to encrypt, authenticate, and digitally sign the information. This can impact the response time and throughput of the protocol.
- Cost: The use of encryption, authentication, and digital signatures can increase the cost of implementing the protocol, particularly for legacy devices that may require hardware upgrades.



```
▸ Frame 1: 255 bytes on wire (2040 bits), 255 bytes captured (2040 bits) on interface 0
▸ Ethernet II, Src: HewlettP_c5:77:a1 (a0:b3:cc:c5:77:a1), Dst: NECPerso_8a:1f:2f (fc:61:98:8a:1f:2f)
▾ Internet Protocol Version 4, Src: 239.191.105.186, Dst: 192.168.1.3
     0100 .... = Version: 4
     .... 0101 = Header Length: 20 bytes (5)
   ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
     Total Length: 237
     Identification: 0x6ba1 (27553)
   ▸ Flags: 0x0000
     Time to live: 128
     Protocol: UDP (17)
     Header checksum: 0x0000 [validation disabled]
     [Header checksum status: Unverified]
     Source: 239.191.105.186
     Destination: 192.168.1.3
▾ User Datagram Protocol, Src Port: 56017, Dst Port: 102
     Source Port: 56017
     Destination Port: 102
     Length: 217
     Checksum: 0xd73b [unverified]
     [Checksum Status: Unverified]
     [Stream index: 0]
▾ Data (209 bytes)
     Data: 0140a3504300f38f23b53e07349975d8be6f7c66486ca9e0...
     [Length: 209]
```

```
0000  fc 61 98 8a 1f 2f a0 b3  cc c5 77 a1 08 00 45 00   ·a···/·· ··w···E·
0010  00 ed 6b a1 00 00 80 11  00 00 ef bf 69 ba c0 a8   ··k····· ····i···
0020  01 03 da d1 00 66 00 d9  d7 3b 01 40 a3 50 43 00   ·····f·· ·;·@·PC·
0030  f3 8f 23 b5 3e 07 34 99  75 d8 be 6f 7c 66 48 6c   ··#·>·4· u··o|fHl
0040  a9 e0 7e 8e 7f 3f 16 02  31 7f 28 4e dc 48 ff 26   ··~··?·· 1·(N·H·&
0050  99 d5 28 a7 22 54 0e 0a  19 93 a4 b7 56 52 ad c9   ··(·"T·· ····VR··
0060  68 3c 82 fd af 30 af 7d  48 e8 8b 8e 9b 7c f9 1c   h<···0·} H····|··
0070  27 0b 05 b8 2b f7 8d f2  4f 0d ab 4d 57 da 29 0b   '···+··· O··MW·)·
0080  7e 75 24 24 9a 7c aa 7a  38 7c 1c aa 8f 39 a8 e3   ~u$$·|·z 8|···9··
0090  74 e5 bb 2f 6a 9e 39 b0  32 9d 56 7e a8 83 e3 f8   t··/j·9· 2·V~····
00a0  eb 2e 97 d8 04 a0 4e 56  5d c6 c2 bb 5b 26 7c 5e   ·.····NV ]···[&|^
00b0  91 41 97 08 15 30 6d 8e  5f 0e 90 13 c4 8f f9 60   ·A···0m· _······`
00c0  66 84 08 30 d9 fb ba 97  44 c1 be a3 ad a6 37 75   f··0···· D·····7u
00d0  18 f6 db d8 3e 4b aa 82  21 85 20 94 86 5b 96 bf   ····>K·· !· ··[··
00e0  c8 8d 25 29 d1 15 24 d2  2b cb 62 58 a5 61 9a 81   ··%)··$· +·bX·a··
00f0  91 99 25 06 88 0b 96 fa  9c 6a 8c ef 1b 13 81      ··%····· ·j·····
```

Figure 4.15: Wireshark frame of secure R-GOOSE

In short, the GoSV tool is able to create custom GOOSE and SV messages. These can be secured with S-GoSV and made routable with R-GoSV. The developed R-GoSV incorporates the specified security methods. The HMAC-SHA256 digital signature and AES-128 encryption techniques are employed to ensure message authentication and integrity and to achieve confidentiality, respectively. Data integrity attacks can be prevented by applying encryption and message authentication techniques, which protect the grid. The proposed method employs authenticated encryption algorithms to mitigate cyberattacks on substations. The use of AES ensures the confidentiality of transmitted data, encompassing protection function operations and status information. This prevents interception and comprehension of exchanged messages by potential attackers. HMAC validates message integrity during transmission, confirming its origin from the expected sender and deterring unauthorised intruders from posing as legitimate devices within the substation network. Authenticated encryption guarantees data integrity, rendering any unauthorised modifications detectable. The failure of decryption or integrity checks signifies potential tampering, thus impeding attackers from modifying data without detection. Even if encrypted data is captured, decryption without the appropriate keys remains unattainable for cyberattacks.

4.5 Conclusion

Chapter 4 presents an overview of TCP client/server communication. The context and outline for the research project are established. The design and implementation of the communication model are studied for requirements of the IEC 61850 protocol. Communication architectures must be designed to meet the unique behaviour and communication requirements of the IEC 61850 protocol. While the standard supports both TCP and UDP communication protocols, TCP is often the preferred choice due to its reliability and error-checking capabilities. However, there are limitations of TCP communication that can impact its effectiveness in IEC 61850 implementation. One of the primary limitations of TCP communication is latency. TCP is a reliable protocol that requires error-checking and acknowledgement of every packet. This can lead to delays in the delivery of data, which can impact the performance of real-time systems such as those found in substation automation. The latency introduced by TCP can result in a delay in the processing of critical events, leading to potential system failures. UDP is often used in IEC 61850 implementation due to its low latency and high speed. However, there are limitations to UDP communication that can impact its effectiveness in substation automation systems. UDP is an unreliable protocol, meaning that it does not provide any error-checking or retransmission mechanisms. This can result in lost or corrupted data packets, which can impact the overall reliability of the system. In substation automation systems, where real-time data transmission is critical, the loss of data can lead to potential safety issues.

An unsecure plain GOOSE and secure GOOSE program is also studied in Chapter 4. Non-secure GOOSE and secure GOOSE are two different approaches to transmitting GOOSE messages with different levels of security. Non-secure GOOSE messages are transmitted without any security mechanisms, while secure GOOSE messages are transmitted with additional security features such as encryption and authentication.

One of the key advantages of non-secure GOOSE is its simplicity. Non-secure GOOSE messages are easy to implement and require minimal processing, making them a cost-effective solution for many applications. Additionally, non-secure GOOSE messages have lower latency, which is important for fast communication in critical systems.

However, non-secure GOOSE has some significant limitations in terms of security. Non-secure GOOSE messages can be intercepted, modified, or even blocked by attackers, which can lead to system failures or data breaches. Therefore, non-secure GOOSE is not suitable for critical systems where the integrity and confidentiality of the data are essential.

Secure GOOSE, on the other hand, provides a higher level of security compared to non-secure GOOSE. Secure GOOSE messages are transmitted with additional security features such as encryption and authentication, which protect the data from interception, modification, and tampering. This makes secure GOOSE suitable for critical systems where the integrity and confidentiality of the data are essential.

However, secure GOOSE has some limitations as well. The additional security features of secure GOOSE may introduce additional processing delays, which can increase latency and reduce the performance of the system. Additionally, the implementation of secure GOOSE can be more complex and costly compared to non-secure GOOSE, which may not be feasible for all applications.

As such, developed security algorithms are to meet data and sampling speeds for performance and security requirements. Developed algorithms that publish GOOSE and R-SV communication must comply with IEC 61850 and implemented authentication and integrity levels must conform to IEC 62351-6 as recommended. According to existing research (Elbez et al., 2019; Farooq et al., 2019), it has been found that RSA-based digital signatures cannot meet the time-critical requirements for communication with substation automation. As an alternative and to meet the desired requirement, the HMAC-based digital algorithm HMAC-SHA26 is implemented to secure GOOSE and R-SV messages in the communications network for power system automation. Careful consideration must be given to the design and implementation of the system to ensure reliability, security, and performance.

Chapter 5 presents a simulation test conducted to evaluate the R-SV communication protocol resulting from the integration of the authentication algorithm. Chapter 5 details the conformance test of the R-SV message structure to the IEC 61850-9-2. Chapter 5 also details the test conducted using Wireshark and Ettercap to verify the frame structure of the R-SV protocol and introduce spoofing network traffic, respectively. A brief description of the research design is presented accordingly:

1. Design Objective: Evaluate current security solutions for GOOSE and R-SV messages. Development of an authentication algorithm for R-SV messages.
2. Experimentation Procedure: The design phase will focus on code development. The implementation phase is the deployment of the presented authenticated encryption algorithm.
3. Analysis: Wireshark is used to collect and analyse data, and Ettercap is used to launch MITM attacks.
4. Interpretation of Results: The Wireshark capture will show all the required fields according to IEC 61850-9-2 for R-SV and the generated attacks of the packets are analysed.

5. CHAPTER FIVE: CASE STUDY IMPLEMENTATION: TESTING AND RESULTS ANALYSIS

5.1 Introduction

The IEC 61850 standard for substation automation systems is widely utilised. While electrical substations continue to digitalise, GOOSE and SV-based line protection applications are becoming more common. The same standard governs IEC 61850 signal exchange between IEDs with different protocols. The conventional approach to protective applications increases the complexity of the protection system. Even with today's IEC 61850 architecture, employing GOOSE and SV-based communications for line distance and differential protection applications would result in significant simplification. For example, with the traditional way of line distance protection, the efficiency of the overall solution is restricted by the speed of the binary contacts, which have relatively slow switching operation times, resulting in longer fault-clearing periods than necessary. As a result of the growing popularity of digital substations, with their undeniable benefits in terms of visibility and flexibility, the use of GOOSE and SV-based protection applications is advantageous, if they meet the required performance parameters, such as dependability and security, as defined by the IEC 62351 and IEC 61850 standards.

This project explores and evaluates the application of standard IEC 61850 and IEC 62351 technology. The implementation aims to develop an encryption and authentication algorithm for IEC 61850 R-SV messages. Based on the literature review, the majority of the authors tend to emphasise security development for the GOOSE protocol. Although R-SV communications are vulnerable to cyberattacks and have a similar frame format to GOOSE messages, they have not attracted the same level of scrutiny as GOOSE messages in the literature. As such, this has led the thesis to develop an algorithm to secure R-SV communication adhere to the frame structure and address this gap. Some solutions in the literature do not enable real-time traffic and thus are incompatible with GOOSE and SV messages. Whereas others violate the required delay of 3 ms or fail to meet the standards, our design authenticates and encrypts IEC 61850 and satisfies IEC 62351:2020. The proposed mechanism in this thesis uses the IEC 62351-6 recommended authentication value extension and encryption of SV APDU in R-SV messages to counteract cyber-attacks.

The following sections detail the project's implementation and present the results of the project. In this chapter, the implementation of the developed authentication code can be used to publish secure sampled value messages as discussed in Section 5.2. Encryption then an

Authentication algorithm is employed on the SV PDU as detailed in Section 5.3 and Section 5.4, respectively. The sampled value messages published from the two devices (Kali-Linux virtual machines) are captured using a commercial protocol analyser tool called Wireshark as illustrated in Section 5.5 to validate the structure of R-SV messages published against that defined in the IEC 61850-9-2 standard. Furthermore, Ettercap will be used to perform an MITM attack by spoofing network traffic as described in Section 5.6.

5.2 Security of Sampled Values

5.2.1 Potential Threats and Vulnerabilities

Because of the nature of the electrical transmission system, the vast majority of substations are unattended. Furthermore, gateways and large networks connect substations to a control centre. Remote access to substations is therefore necessary. The most serious concern with remote access is that appropriate security controls may not be implemented. As a result, substation infiltration could happen in several ways. An attacker may infect a laptop connected to the substation communication network. As a result, if equipment located at the lower level is hacked, the intruder could be able to exploit the process bus network.

Because of its features, such as plain text messages and multicast at the data link layer, the SV protocol reveals all data details within a network. An attacker with process bus access can inspect the R-SV message. As a result, launch a cyber-attack by changing voltage measurements. Another method for compromising the R-SV message is to exploit weaknesses in the subscriber's processing protocol.

5.2.1.1 Replay Attacks and Masquerade Attacks

The intruder can initiate two types of cyberattacks against SVs: replay attacks and masquerade attacks. During regular operation, an intruder intercepts a substation network and records an R-SV message packet including current and voltage measurements. During a malfunction, the intruder then replicates the intercepted packets into the substation network. During the failure, the IED receives normal current and voltage parameters. This would cause the IED to maintain the circuit breaker closed irrespective presence of a malfunction. The fault level could surge, and the network could potentially fail. This would cause significant damage to equipment, and power outages, and potentially jeopardise the reliability of power. In a masquerade attack, an attacker gains access to the substation network by establishing a false identity and retrieving and intercepting an R-SV communication packet to manipulate or alter the measured values. The initial R-SV packet would be completely tampered with as a result. A malicious attacker may manipulate the R-SV data, causing the IED process of operation to produce unexpected results, which could have negative consequences.

Overall, the standard IEC 61850-9-2 SV is proven to be sensitive to replay and masquerading attacks (Suhail Hussain et al., 2023). As a result, the presented technique in this study incorporates the IEC 62351-6 recommended authentication value extension and encryption of SV frame structure in R-SV communications to mitigate cyber-attacks.

5.2.2 Sampled Measured Values

The R-SV message is used to communicate measured samples from sensor systems between IED devices. The R-SV message is a multicast technique that is used for data transmission between several IEDs connected to an Ethernet network. The OSI model layer 2 is utilised for mapping R-SV data. The bottom tiers of the ISO/OSI model use Ethernet multicast and serial line unicast communication. As previously discussed in Section 3.3.2, the following fields are contained within the SV packet frame as discussed by (Karnati, 2020):

- *“Destination address*
- *Source address: The address of the publisher.*
- *VLAN priority tag: Priority tagging according to IEEE 802.1Q.*
- *Ethertype: SV Ethertype is set to 88-BA.*
- *APPID: Application identifier.*
- *Length: The total number of bytes in the SV message.*
- *Reserved 1: Reserved for future standardisation.*
- *Reserved 2: Reserved for future standardisation.*
- *APDU: APDU contains SV data structure.”*

The information to be distributed in the process bus network is encoded in the SV buffer as an APDU. The following fields are contained in the APDU of the SV packet as discussed by (Karnati, 2020):

- *“svID: Should be a system-wide unique identification.*
- *smpCnt: Each time a new sampling value is taken, this value will be incremented. If the sample is synchronised by a clock signal and the synchronising signal occurs, the counter must be set to zero.*

- *ConfRef: Value from the MSVCB.*
- *RefrTm: Contains the refresh time of the SV buffer.*
- *smpSynch: Synchronised by an external clock signal.*
- *seqData: List of data values related to the data set definition."*

5.2.2.1 Replay Attacks in IEC 61850 GOOSE and Sampled Value Messages Comparison

Replay attacks in GOOSE communication are easily identified by evaluating the values of the *stNum* and *sqNum* fields of the incoming GOOSE with the last received GOOSE communication. With each additional GOOSE communication, the *sqNum* value is increased. Whereas the *stNum* value is increased whenever there is an event in data set information. When the *stNum* is increased, the *sqNum* value is reset to 0. As a result, any replay attack in GOOSE messages can be easily discovered by comparing the current GOOSE communication's *stNum* and *sqNum* values to the old GOOSE communication's *stNum* and *sqNum* values. The *smpCnt* field in SV messages is incremented for each new SV message and its value is reset to 0 every second. The SV message's *smpCnt* value is equivalent to the GOOSE message's *sqNum* value. However, the SV message lacks a value similar to *stNum* in the GOOSE message. Because the *smpCnt* value is reset every second, it is insufficient to detect replay attacks in SV messages. The optional field Security in the IEC 61850-9-2 SV APDU is kept for future specification and use. The security field is used to store the timestamp, or the moment at which the SV frame was formatted. In the proposed security method, the security field timestamp value, coupled with the *smpCnt* value, is used to detect replay attacks in SV messages. With the proposed security approach, the Security field of each SV packet now comprises the time at which the packet was produced. If the received SV packet's timestamp value is less than or equal to the last received timestamp, replay is detected, and the packet is deleted. If the value is greater, the SV packet is processed further, and the last received timestamp value is updated with the current SV packet's timestamp value. (Suhail Hussain et al., 2023)

5.3 Message Authentication Code

5.3.1 Hash Message Authentication Code (HMAC)

As an authentication value, IEC 62351-6:2007 proposes using RSA-based digital signatures. The processing time for RSA-based digital signatures is 2-3 msec. As a result, it is unsuitable for SVs. Conversely, because MAC methods have relatively short processing durations, they can be utilised to generate this authentication value. Several MAC methods recommended in the recently published IEC 62351-6:2020 are employed to generate the authentication value in the proposed mechanism. Additionally, the SV APDU is encrypted using Advanced Encryption Standard using the HMAC SHA-256 method.

Validating the integrity of data carried over or kept on an unprotected medium is critical in the age of open network communications. Message Authentication Codes (MACs) are commonly used to validate data exchanged between two parties who share a secret key.

To authenticate a plaintext communication message, a secure tag, such as MAC, is utilised. The MAC can be created from the original message and includes limited procedures to confirm that the publisher has not tampered with the message's integrity. Producing MAC from the message context, on the other hand, will require processing power and time, which is a key concern for the real-time operation of the substation automation system because SV must come within 3ms, according to IEC 61850. As a result, the SV message must take precedence over other communication messages, and encryption methods are not advised due to the additional processing time required (Karnati, 2020).

HMAC is a hash function-based algorithm that can assure message integrity and authentication. The following are some of the benefits of HMAC:

- the tag's length is short and fixed,
- preventing duplication,
- the original message is hidden.

It is challenging to calculate the identical inputs from the output HMAC tags due to the collision resistance and one-way function properties (Refer to Section 2.3, Equation 2.1). As a result, a secret key is usually used by HMAC. The fundamental goals of this construction are as follows:

- To utilise hash functions in their original form.
- To retain the original hash function output without incurring major degradation.
- To give a well-understood cryptographic assessment of the authentication mechanism's strength.

5.4 Authenticated Encryption

The most common use of cryptography on the Internet these days is to create a secure channel between two endpoints and then exchange data over that channel. Typical implementations initiate a key-exchange protocol to establish a shared key between the participants, and then utilise this key to authenticate and encrypt the transmitted data using efficient symmetric key algorithms as depicted in Figure 5.1. Usually, when encryption is employed on data, a MAC is required to provide additional security, since encryption alone is insufficient to protect data from intruders. An authorised encryption mode of operation must provide both privacy and communication authenticity.

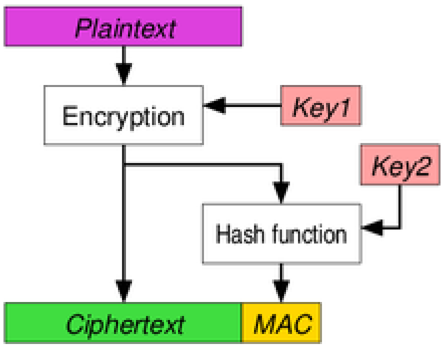


Figure 5.1: EtM security algorithm

Hash or message authentication mechanisms are utilised to ensure the integrity of communication. Encryption and integrity techniques are sometimes used together:

- Encrypt-then-MAC (EtM): ensures ciphertext confidentiality, but no plaintext confidentiality,
- MAC-then-encrypt (MtE): ensures plaintext confidentiality, but no ciphertext confidentiality, and
- Encrypt-and-MAC (EandM): ensures plaintext confidentiality, but no ciphertext confidentiality.

Figure 5.1 illustrates the EtM algorithm. The encryption technique used by the sender returns a ciphertext (including the authentication tag), but the decryption process used by the receiver returns either a plaintext or a special symbol indicating that the ciphertext is faulty or unauthentic. To construct a fresh digest if the user tampers with the ciphertext, they must also know the HMAC key. The ciphertext will not authenticate if the user modifies the digest (Oszywa & Gliwa, 2012). This is advantageous for two reasons: first, it makes a denial-of-service assault considerably more difficult by allowing you to discard faked packets more quickly, and second, it limits your "attack surface."

EtM is the most secure algorithm since modifications to the ciphertext can be filtered out prior to decryption using a valid MAC code, preventing communications tampering. MtE and EandM each offer varying levels of security, but EtM offers the whole package. The user cannot tamper with the ciphertext when EtM is used.

Encryption hides your data but does not prevent tampering. As a result, authentication has been incorporated into the algorithm. Thus, establishing, and standardising specialised authenticated encryption techniques is quite valuable. The EtM technique is the best for achieving authenticated encryption and should minimise or remove a variety of attacks on the present MtE mechanism. (Gutmann, 2014)

5.5 Implementation

For the IEC 61850 message exchange, the IEC 62351-6:2020 standard outlines the use of security requirements such as digital signatures that use RSA algorithms to ensure integrity and authenticity. However, it makes no provision for protecting the confidentiality of R-SV messages. With the rapid expansion of IEC 61850 from substation automation to power management, IEC 61850 messages are now being used to transmit sensitive data that requires secrecy. Previous research has found that when applied to R-SV messages, the generation of digital signatures using RSA and ECDSA algorithms requires long computational and processing times (Harispuru & Schuster, n.d.); Gonzalez-Redondo et al., 2013; Firouzi et al., 2017). Because R-SV communications have a 3 ms time limit, IEC 62351-1 advises against using encryption methods. However, R-SV messages must be encrypted to achieve the confidentiality requirement. Moreover, the implementation of encryption algorithms must conform to the timing restriction of 3ms. Most of the literature does not consider security mechanisms for R-SV messages, especially confidentiality requirements.

In this chapter, the iterative implementation procedures are documented, to address the lack of information. A method is proposed to ensure the confidentiality and message authentication of R-SV messages by employing Authenticated Encryption with Associated Data (AEAD) algorithms. Further, C-library-based implementations are developed by programming R-SV data frames according to the IEC 61850-9-2 standard to test the timing performance and feasibility of the proposed security method for R-SV messages. The complete source code of the interface is included in Annexure E and Annexure F.

5.5.1 Proposed Method for Achieving Confidentiality in SV Messages

In the R-SV protocol, the suggested method uses the EtM variant of the AEAD algorithms to employ confidentiality, message integrity, and authenticity.

The security scheme for R-SV messages in IEC 61850 was originally based on MACs and digital signatures to ensure the integrity and authenticity of the data. However, this scheme had some limitations and was vulnerable to various attacks such as replay attacks, man-in-the-middle attacks, and message insertion attacks. Therefore, the security scheme is to address these vulnerabilities and improve the overall security of R-SV communication.

The adapted security scheme for R-SV communication in IEC 61850 includes the following measures:

1. Encryption: Encryption is used to protect the confidentiality of R-SV messages by ensuring that only authorised devices can access the data. Encryption algorithms such as AES-128 are used to encrypt the messages before transmission over the network. However, as research progresses, new attack methods such as the Biclique attack emerge. Attackers are aware that in order to carry out a brute force assault against AES-128, they must try every possible key combination until the correct key is identified. A biclique attack is a type of attack that exploits the block cipher's algebraic structure. It specifically checks for collisions throughout the encryption process. A biclique attack focuses on discovering collisions in the encryption mechanism, possibly decreasing the complexity of the attack. The biclique attack is currently the only key-recovery attack on complete AES using a single key. (Bogdanov et al., 2011b) used it to build bicliques for all three versions of AES. The biclique attack is approximately four times faster than brute force. The biclique assault can only be implemented hypothetically. As a result, there are no practical implications for AES encryption. There is currently no known viable attack that would allow someone who does not know the key to access data encrypted by AES when properly implemented.
2. Timestamping: Timestamping is used to prevent replay attacks by adding a timestamp to each message. The receiver can then check the timestamp to ensure that the message is fresh and has not been replayed.
3. Sequence numbers: Sequence numbers are used to prevent message insertion attacks by assigning a unique sequence number to each message. The receiver can then check the sequence number to ensure that the message is in the correct order and has not been inserted or modified.

4. HMAC: HMACs are used to ensure the integrity of R-SV messages by generating a unique signature for each message. The signature is based on a hash function and a secret key that is shared between the sender and receiver. Having more than one HMAC with different keys or algorithms can enhance security however there are potential risks. Managing multiple HMACs can increase complexity in key management. This complexity could potentially introduce vulnerabilities. If there are implementation flaws, they could be exploited regardless of the number of HMACs used.
5. For AES-HMAC to work securely, both the sender and receiver must agree on a shared secret key and specific algorithm used including the AES variant and HMAC function. The structure of the message being exchanged must be well-defined.

The EtM send program is responsible for constructing an Ethernet frame that includes all the necessary fields of the frame, including the destination address, the source address, the ether type, and then the SV PDU fields. As illustrated in Figure 5.2, the fields that make up the PDU comprise APPID, the length of the PDU, Reserved1, Reserved2, and SV APDU fields, and then they are followed by the Frame Check Sequence (FCS) field.

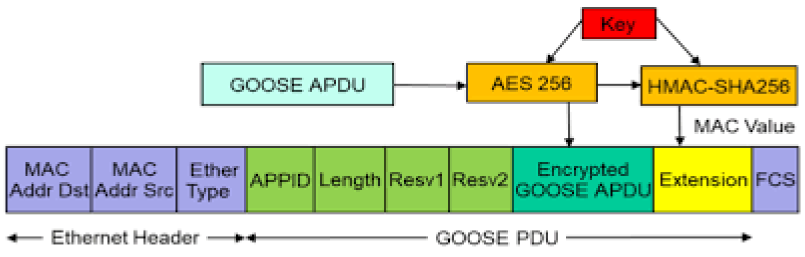


Figure 5.2: EtM algorithm applied to GOOSE or SV PDU (Hussain, Farooq, et al., 2020)

In the event that the frame is secured via a digital signature mechanism, the Reserved1 field will contain the length of the Extension field. There is no extension field for SV messages when its value is 0, which indicates that no security is being applied to such communications. The authentication value (MAC value) for SV PDU is derived from the Ethertype field till the end of the encrypted SV APDU. The length of the Extension field appended to the SV PDU is added to the second byte of the SV PDU's reserved1 field. The Cyclic Redundancy Check (CRC) value is stored in the Reserved2 column. The Tag-Length-Value (TLV) format is used for the fields that make up the SV APDU. While processing the SV frames at the publisher, the SV APDU is encrypted first, and then the MAC value is created and added to the extension field. The EtM transmit software uses AES-128 encryption to encrypt the SV APDU after it has completed the construction of the SV ethernet frame. The total amount of space used up by the SV APDU fields is 137 bytes. Padding brings the total size of an SV APDU up to 144 bytes, which is necessary because AES-128 encrypts 16 bytes of data at a time. The cipher text that is produced as the output of AES-128 is then passed as an input to the HMAC-SHA-256 generating function, which produces a MAC value consisting of 32 bytes. The MAC value that has been created is saved in an extended field of the SV APDU. The extension fields include the version, Time of Current Key, Time of Next Key, Security Algorithm, and Key ID, followed by the 32-byte MAC value, as shown in Figure 5.3. The most significant byte is the one that is used to represent the encryption technique, while the least significant byte is the one that is used to indicate the authentication algorithm for the message (Ustun et al., 2020). Every second, the EtM send program constructs the SV PDU frame in a buffer character array and sends it to the network. The times of encryption and MAC creation are captured. The publisher sends the secure SV packets.

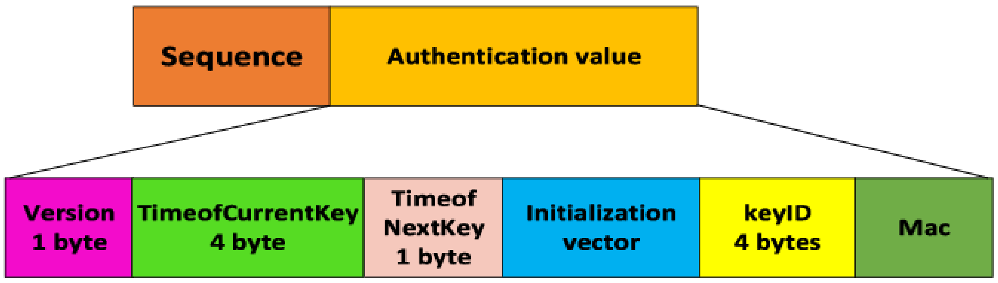


Figure 5.3: Structure of Extension field (Hussain, Farooq, et al., 2020)

Upon receiving the secure R-SV packet, the subscriber obtains the MAC value in the extension field and stores it. In the EtM receiver program, which receives data from the sender program, the cipher text and the MAC value are extracted from the received data into cipher and hash arrays. In addition, the EtM receiver software compares the received MAC value with a newly generated MAC value using the HMAC-SHA-256 creation function. The encrypted text will be decrypted if the two values match; otherwise, the packet will not be further processed. As a result, the time it takes to generate and decode the MAC is recorded.

5.5.2 Implementation and Performance Evaluation

To implement an IEC 61850 communication system, a thorough grasp of the processes, techniques, and technologies is required. In addition, a new framework for substation communication based on the IEC 61850 standard is established to solve the cybersecurity challenges stated in IEC 62351 for critical infrastructure communication.

The proposed approach for R-SV security, which makes use of the EtM algorithm, has been put into action. Simulation is implemented to examine the reliability of securing R-SV communications and to determine whether it is feasible. Furthermore, the simulation confirms the SV PDU as stated in Part 9-2 of the IEC 61850 standard. The message structure is checked by simulating and recording the R-SV message using network protocol analyser software, followed by packet frame analysis. Figure 5.4 depicts the experimental setup for R-SV message simulation and validation, with the EtM transmitter and receiver programs running on Kali-Linux software installed on a personal computer (PC). To collect R-SV message packets and perform MITM attacks, Wireshark and Ettercap software running on a PC are utilised. A network switch connects the two machines. The developed security system satisfies the requirements for message confidentiality, integrity, and authentication. Also, the recommended security technique works well to protect against unauthorised access, spoofing, and MITM attacks.

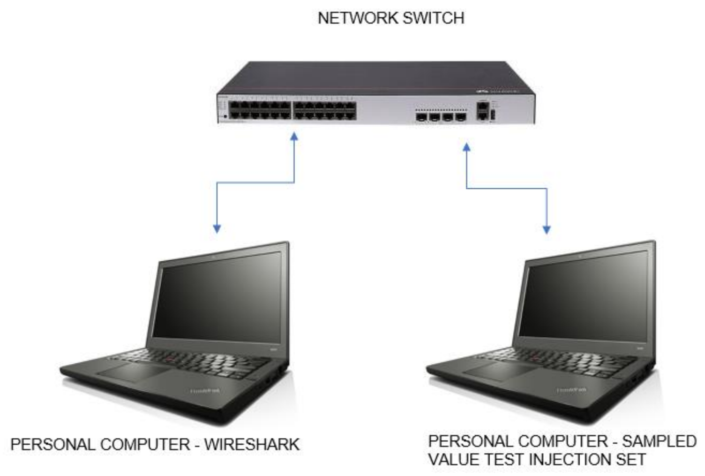
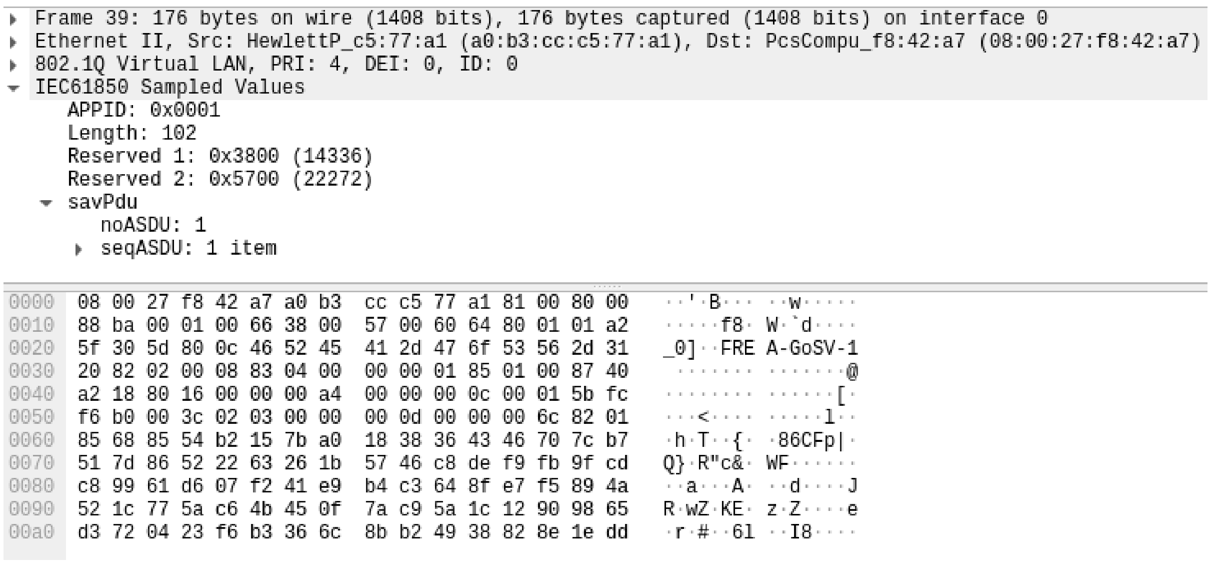

NETWORK SWITCH
PERSONAL COMPUTER - WIRESHARK
PERSONAL COMPUTER - SAMPLED VALUE TEST INJECTION SET

Figure 5.4: PC with Kali-Linux and PC with Wireshark Network Analyzer Software and Ettercap

EtM is utilised to secure R-SV communications transmitted between the publisher and the subscriber. Figure 5.3 depicts the overall security process of the proposed mechanism, while Table 5.1 provides further information about the suggested security protocol. Using AES-128 encryption and HMAC SHA-256 message integrity, the EtM algorithm safeguards R-SV message communication. The publisher and subscriber programs are executed on two distinct terminals that function as two IEDs. As shown in Table 5.1, the publisher terminal initiates the R-SV message creation process, Gen_EtM(), which generates a secure R-SV message including an encrypted SV APDU (E_d) and MAC value “h” using the symmetric PreSharedKey “k”. Figure 5.5 depicts the Wireshark capture of an encrypted and authenticated R-SV message broadcast by the EtM AEAD algorithm of the S-GoSV library.



```
▸ Frame 39: 176 bytes on wire (1408 bits), 176 bytes captured (1408 bits) on interface 0
▸ Ethernet II, Src: HewlettP_c5:77:a1 (a0:b3:cc:c5:77:a1), Dst: PcsCompu_f8:42:a7 (08:00:27:f8:42:a7)
▸ 802.1Q Virtual LAN, PRI: 4, DEI: 0, ID: 0
▾ IEC61850 Sampled Values
     APPID: 0x0001
     Length: 102
     Reserved 1: 0x3800 (14336)
     Reserved 2: 0x5700 (22272)
   ▾ savPdu
        noASDU: 1
      ▸ seqASDU: 1 item

0000  08 00 27 f8 42 a7 a0 b3  cc c5 77 a1 81 00 80 00   ··'·B···  ··w·····
0010  88 ba 00 01 00 66 38 00  57 00 60 64 80 01 01 a2   ·····f8·  W·`d····
0020  5f 30 5d 80 0c 46 52 45  41 2d 47 6f 53 56 2d 31   _0]··FRE  A-GoSV-1
0030  20 82 02 00 08 83 04 00  00 00 01 85 01 00 87 40    ·······  ·······@
0040  a2 18 80 16 00 00 00 a4  00 00 00 0c 00 01 5b fc   ········  ······[·
0050  f6 b0 00 3c 02 03 00 00  00 0d 00 00 00 6c 82 01   ···<····  ·····l··
0060  85 68 85 54 b2 15 7b a0  18 38 36 43 46 70 7c b7   ·h·T··{·  ·86CFp|·
0070  51 7d 86 52 22 63 26 1b  57 46 c8 de f9 fb 9f cd   Q}·R"c&·  WF······
0080  c8 99 61 d6 07 f2 41 e9  b4 c3 64 8f e7 f5 89 4a   ··a···A·  ··d····J
0090  52 1c 77 5a c6 4b 45 0f  7a c9 5a 1c 12 90 98 65   R·wZ·KE·  z·Z····e
00a0  d3 72 04 23 f6 b3 36 6c  8b b2 49 38 82 8e 1e dd   ·r·#··6l  ··I8····
```

Figure 5.5: Packet capture of EtM SV message

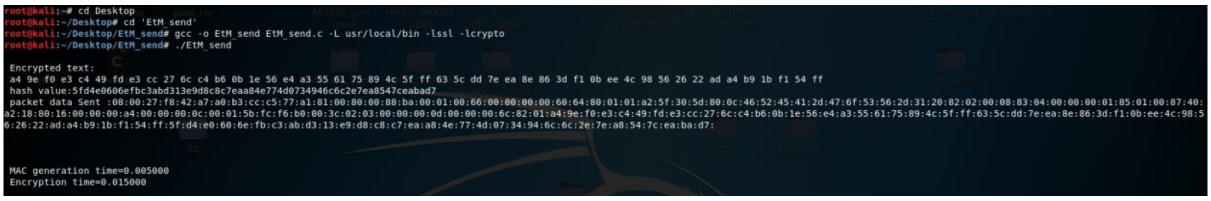
The subscriber reads the encrypted APDU (E_d) values into the ReceivedData buffer after receiving the secure R-SV message. As shown in Table 5.1, the procedure verify_EtM() is utilised to validate the R-SV message's integrity. First, a new MAC value "h1" is generated using the symmetric PreSharedKey "k" for the received encrypted SV APDU (E_d). "h1" is compared to the received MAC value "h" and validated. The encrypted SV PDU is decrypted and processed further if they match; otherwise, it is refused since the MAC value mismatch signifies that at least one of the received encrypted SV PDUs or MAC values has been altered. To obtain the APDU data for an approved packet, the encrypted SV APDU (E_d) is decrypted using the key "k."

Table 5.1: Publisher and Subscriber Algorithm

Algorithm Gen_EtM( )	**Algorithm verify_EtM()**
svPDU ← GoSV()	ReceivedData ← svPDU.APDU = E_d
InputData ← svPDU.APDU	h ← svPDU.extension
k ← PreSharedKey()	k ← PreSharedKey()
E_d ← EncryptK (InputData)	h1 ← MACk (ReceivedData)
h ← MACk (E_d)	if h = h1 then
svPDU.Extension ← h	APDU ← DecryptK (E_d)
svPDU.APDU ← Ed	else
	return “Reject SV packet”

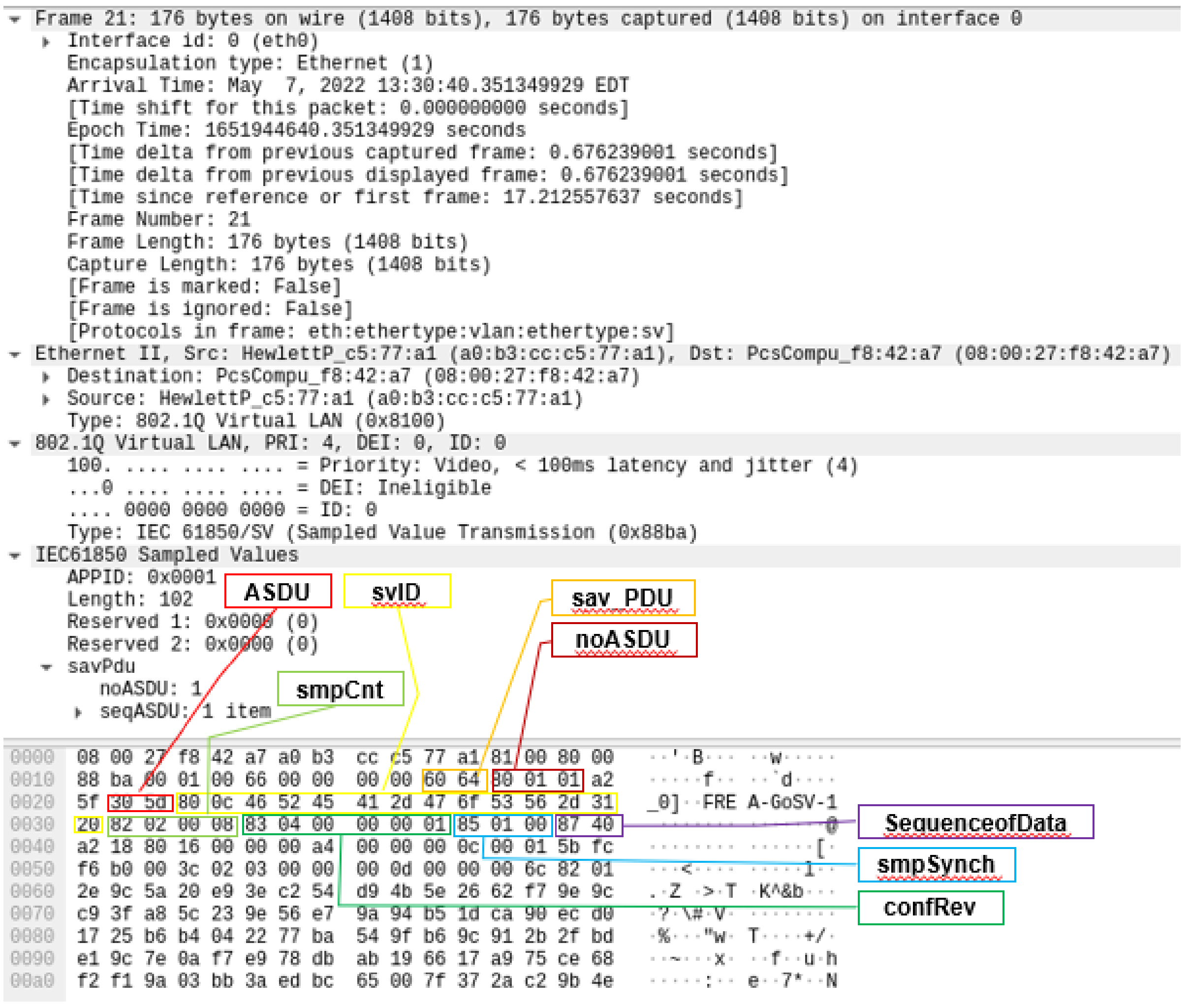
To evaluate timing performance, the computational times required to execute the EtM security version are computed. The testing system is an Intel(R) Core(TM) i7-10510U with 16GB RAM running Kali-Linux with EtM sender and receiver software with GCC compiler. Compile and run the application at the command prompt. The EtM sender program execution steps are as follows:

- Install libssl library.
- Replace the destination and source MAC addresses with your intended MAC addresses in the EtM_send.c program.
- Replace the network interface name with your computer’s network interface name in the EtM_send.c program.
- Then compile and run the program as illustrated in Figure 5.6.
- The server code is responsible for providing the data to the clients. This involves creating a socket connection and binding it to a server port. The server code then listens for incoming client requests and responds with the requested data.
- Figure 5.7 shows the secure R-SV packet captures of the publisher device employed with the EtM technique. It incorporates all the authentication inputs of the SV PDU. Wireshark shows all the required fields according to IEC 61850-9-2. Refer to Appendix G for more detailed packet capture.



```
root@kali:~# cd Desktop
root@kali:~/Desktop# cd 'EtM_send'
root@kali:~/Desktop/EtM_send# gcc -o EtM_send EtM_send.c -L usr/local/bin -lssl -lcrypto
root@kali:~/Desktop/EtM_send# ./EtM_send

 Encrypted text:
 a4 9e f0 e3 c4 49 fd e3 cc 27 6c c4 b6 0b 1e 56 e4 a3 55 61 75 89 4c 5f ff 63 5c dd 7e ea 8e 86 3d f1 0b ee 4c 98 56 26 22 ad a4 b9 1b f1 54 ff
 hash value:5fd4e0606efbc3abd313e9d8c8c7eaa84e774d0734946c6c2e7ea8547ceabad7
 packet data Sent :08:00:27:f8:42:a7:a0:b3:cc:c5:77:a1:81:00:80:00:88:ba:00:01:00:66:00:00:00:00:60:64:80:01:01:a2:5f:30:5d:80:0c:46:52:45:41:2d:47:6f:53:56:2d:31:20:82:02:00:08:83:04:00:00:00:01:85:01:00:87:40:
a2:18:80:16:00:00:00:a4:00:00:00:0c:00:01:5b:fc:f6:b0:00:3c:02:03:00:00:00:0d:00:00:00:6c:82:01:a4:9e:f0:e3:c4:49:fd:e3:cc:27:6c:c4:b6:0b:1e:56:e4:a3:55:61:75:89:4c:5f:ff:63:5c:dd:7e:ea:8e:86:3d:f1:0b:ee:4c:98:5
6:26:22:ad:a4:b9:1b:f1:54:ff:5f:d4:e0:60:6e:fb:c3:ab:d3:13:e9:d8:c8:c7:ea:a8:4e:77:4d:07:34:94:6c:6c:2e:7e:a8:54:7c:ea:ba:d7:



 MAC generation time=0.005000
 Encryption time=0.015000
```

Figure 5.6: Terminal output for publisher device



```
▾ Frame 21: 176 bytes on wire (1408 bits), 176 bytes captured (1408 bits) on interface 0
  ▸ Interface id: 0 (eth0)
    Encapsulation type: Ethernet (1)
    Arrival Time: May  7, 2022 13:30:40.351349929 EDT
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1651944640.351349929 seconds
    [Time delta from previous captured frame: 0.676239001 seconds]
    [Time delta from previous displayed frame: 0.676239001 seconds]
    [Time since reference or first frame: 17.212557637 seconds]
    Frame Number: 21
    Frame Length: 176 bytes (1408 bits)
    Capture Length: 176 bytes (1408 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:vlan:ethertype:sv]
▾ Ethernet II, Src: HewlettP_c5:77:a1 (a0:b3:cc:c5:77:a1), Dst: PcsCompu_f8:42:a7 (08:00:27:f8:42:a7)
  ▸ Destination: PcsCompu_f8:42:a7 (08:00:27:f8:42:a7)
  ▸ Source: HewlettP_c5:77:a1 (a0:b3:cc:c5:77:a1)
    Type: 802.1Q Virtual LAN (0x8100)
▾ 802.1Q Virtual LAN, PRI: 4, DEI: 0, ID: 0
    100. .... .... .... = Priority: Video, < 100ms latency and jitter (4)
    ...0 .... .... .... = DEI: Ineligible
    .... 0000 0000 0000 = ID: 0
    Type: IEC 61850/SV (Sampled Value Transmission (0x88ba)
▾ IEC61850 Sampled Values
    APPID: 0x0001
    Length: 102
    Reserved 1: 0x0000 (0)
    Reserved 2: 0x0000 (0)
  ▾ savPdu
      noASDU: 1
    ▸ seqASDU: 1 item

0000  08 00 27 f8 42 a7 a0 b3  cc c5 77 a1 81 00 80 00   ··'·B··· ··w·····
0010  88 ba 00 01 00 66 00 00  00 00 60 64 80 01 01 a2   ·····f·· ··`d····
0020  5f 30 5d 80 0c 46 52 45  41 2d 47 6f 53 56 2d 31   _0]··FRE A-GoSV-1
0030  20 82 02 00 08 83 04 00  00 00 01 85 01 00 87 40    ······· ·······@
0040  a2 18 80 16 00 00 00 a4  00 00 00 0c 00 01 5b fc   ········ ······[·
0050  f6 b0 00 3c 02 03 00 00  00 0d 00 00 00 6c 82 01   ···<···· ·····l··
0060  2e 9c 5a 20 e9 3e c2 54  d9 4b 5e 26 62 f7 9e 9c   .·Z ·>·T ·K^&b···
0070  c9 3f a8 5c 23 9e 56 e7  9a 94 b5 1d ca 90 ec d0   ·?·\#·V· ········
0080  17 25 b6 b4 04 22 77 ba  54 9f b6 9c 91 2b 2f bd   ·%···"w· T····+/·
0090  e1 9c 7e 0a f7 e9 78 db  ab 19 66 17 a9 75 ce 68   ··~···x· ··f··u·h
00a0  f2 f1 9a 03 bb 3a ed bc  65 00 7f 37 2a c2 9b 4e   ·····:·· e··7*··N
```

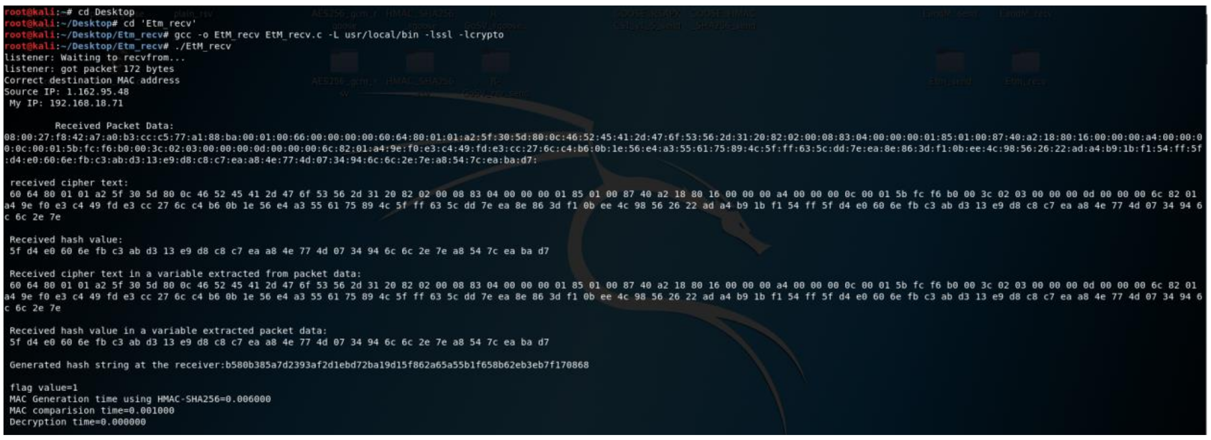
Figure 5.7: Packet Capture of Publisher

The EtM receiver program execution steps are as follows:

- Install libssl library.
- Replace the destination MAC address with your receiver computer's MAC address in the EtM_recv.c program.
- Replace the network interface name with your computer's network interface name in the EtM_recv.c program.
- Then compile and run the program as illustrated in Figure 5.8.

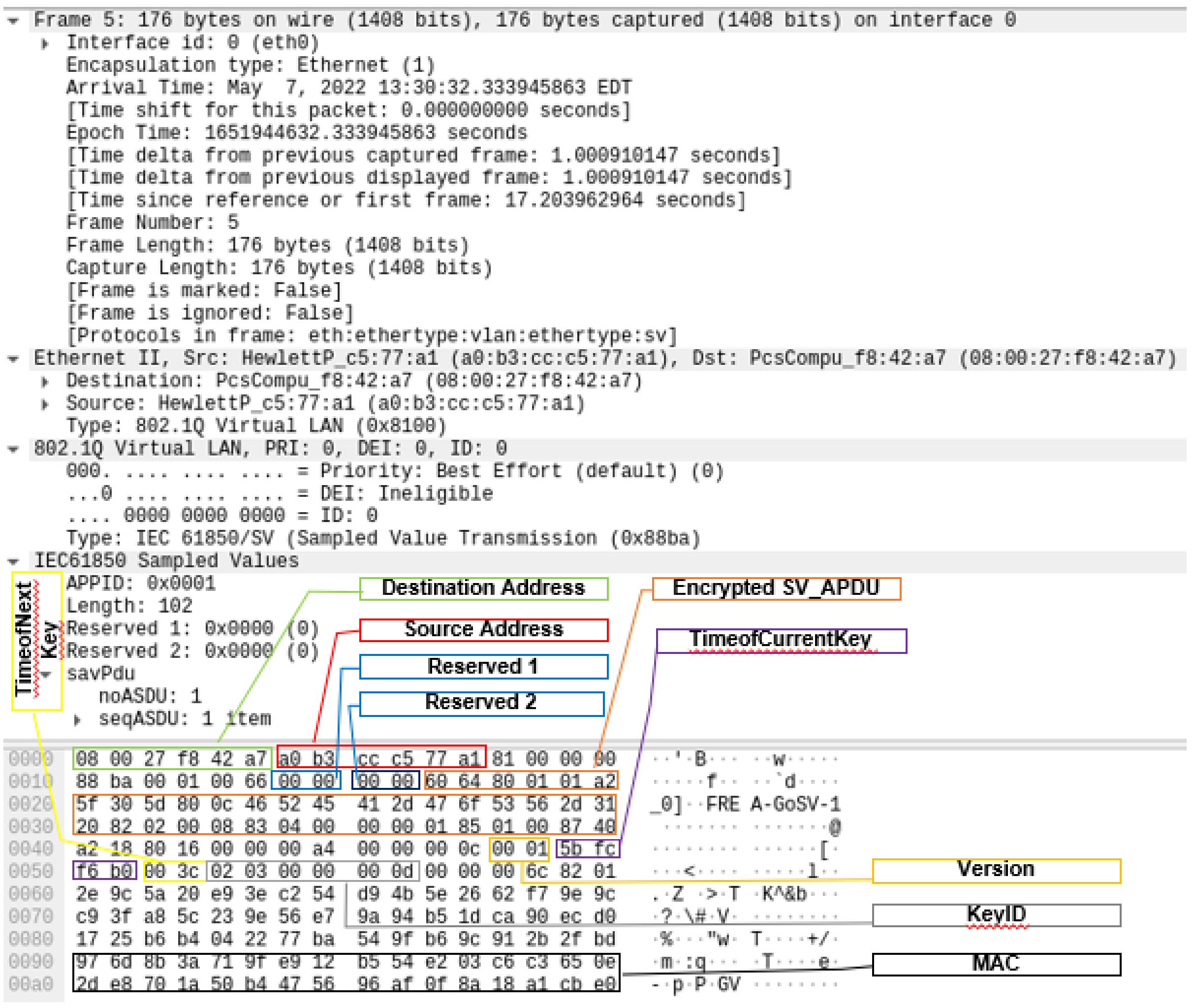
- The client code is responsible for requesting the data from the server. This involves creating a socket connection and sending a request message to the server. The client code then receives the requested data from the server and processes it according to the data model.
- Figure 5.9 shows the secure R-SV packet captures of the subscriber device employed with the EtM algorithm. Wireshark shows all the required fields according to IEC 61850-9-2. Refer to Appendix H for more detailed packet capture.

The final step in developing a publisher-subscriber code algorithm is to validate the data model. This involves checking that the data objects exchanged between publishers and subscribers are consistent with the data model. Any errors or inconsistencies should be logged and handled appropriately. The use of multicast messages reduces the amount of network traffic and improves the efficiency of communication.



```
root@kali:~# cd Desktop
root@kali:~/Desktop# cd 'Etm_recv'
root@kali:~/Desktop/Etm_recv# gcc -o EtM_recv EtM_recv.c -L usr/local/bin -lssl -lcrypto
root@kali:~/Desktop/Etm_recv# ./EtM_recv
listener: Waiting to recvfrom...
listener: got packet 172 bytes
Correct destination MAC address
Source IP: 1.162.95.48
 My IP: 192.168.18.71

         Received Packet Data:
08:00:27:f8:42:a7:a0:b3:cc:c5:77:a1:88:ba:00:01:00:66:00:00:00:00:60:64:80:01:01:a2:5f:30:5d:80:0c:46:52:45:41:2d:47:6f:53:56:2d:31:20:82:02:00:08:83:04:00:00:00:01:85:01:00:87:40:a2:18:80:16:00:00:00:a4:00:00:0
0:0c:00:01:5b:fc:f6:b0:00:3c:02:03:00:00:00:0d:00:00:00:6c:82:01:a4:9e:f0:e3:c4:49:fd:e3:cc:27:6c:c4:b6:0b:1e:56:e4:a3:55:61:75:89:4c:5f:ff:63:5c:dd:7e:ea:8e:86:3d:f1:0b:ee:4c:98:56:26:22:ad:a4:b9:1b:f1:54:ff:5f
:d4:e0:60:6e:fb:c3:ab:d3:13:e9:d8:c8:c7:ea:a8:4e:77:4d:07:34:94:6c:6c:2e:7e:a8:54:7c:ea:ba:d7:

 received cipher text:
 60 64 80 01 01 a2 5f 30 5d 80 0c 46 52 45 41 2d 47 6f 53 56 2d 31 20 82 02 00 08 83 04 00 00 00 01 85 01 00 87 40 a2 18 80 16 00 00 00 a4 00 00 00 0c 00 01 5b fc f6 b0 00 3c 02 03 00 00 00 0d 00 00 00 6c 82 01
a4 9e f0 e3 c4 49 fd e3 cc 27 6c c4 b6 0b 1e 56 e4 a3 55 61 75 89 4c 5f ff 63 5c dd 7e ea 8e 86 3d f1 0b ee 4c 98 56 26 22 ad a4 b9 1b f1 54 ff 5f d4 e0 60 6e fb c3 ab d3 13 e9 d8 c8 c7 ea a8 4e 77 4d 07 34 94 6
c 6c 2e 7e

 Received hash value:
 5f d4 e0 60 6e fb c3 ab d3 13 e9 d8 c8 c7 ea a8 4e 77 4d 07 34 94 6c 6c 2e 7e a8 54 7c ea ba d7

 Received cipher text in a variable extracted from packet data:
 60 64 80 01 01 a2 5f 30 5d 80 0c 46 52 45 41 2d 47 6f 53 56 2d 31 20 82 02 00 08 83 04 00 00 00 01 85 01 00 87 40 a2 18 80 16 00 00 00 a4 00 00 00 0c 00 01 5b fc f6 b0 00 3c 02 03 00 00 00 0d 00 00 00 6c 82 01
a4 9e f0 e3 c4 49 fd e3 cc 27 6c c4 b6 0b 1e 56 e4 a3 55 61 75 89 4c 5f ff 63 5c dd 7e ea 8e 86 3d f1 0b ee 4c 98 56 26 22 ad a4 b9 1b f1 54 ff 5f d4 e0 60 6e fb c3 ab d3 13 e9 d8 c8 c7 ea a8 4e 77 4d 07 34 94 6
c 6c 2e 7e

 Received hash value in a variable extracted packet data:
 5f d4 e0 60 6e fb c3 ab d3 13 e9 d8 c8 c7 ea a8 4e 77 4d 07 34 94 6c 6c 2e 7e a8 54 7c ea ba d7

 Generated hash string at the receiver:b580b385a7d2393af2d1ebd72ba19d15f862a65a55b1f658b62eb3eb7f170868

 flag value=1
 MAC Generation time using HMAC-SHA256=0.006000
 MAC comparision time=0.001000
 Decryption time=0.000000
```

Figure 5.8: Terminal output for subscriber device



```
▾ Frame 5: 176 bytes on wire (1408 bits), 176 bytes captured (1408 bits) on interface 0
  ▸ Interface id: 0 (eth0)
    Encapsulation type: Ethernet (1)
    Arrival Time: May  7, 2022 13:30:32.333945863 EDT
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1651944632.333945863 seconds
    [Time delta from previous captured frame: 1.000910147 seconds]
    [Time delta from previous displayed frame: 1.000910147 seconds]
    [Time since reference or first frame: 17.203962964 seconds]
    Frame Number: 5
    Frame Length: 176 bytes (1408 bits)
    Capture Length: 176 bytes (1408 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:vlan:ethertype:sv]
▾ Ethernet II, Src: HewlettP_c5:77:a1 (a0:b3:cc:c5:77:a1), Dst: PcsCompu_f8:42:a7 (08:00:27:f8:42:a7)
  ▸ Destination: PcsCompu_f8:42:a7 (08:00:27:f8:42:a7)
  ▸ Source: HewlettP_c5:77:a1 (a0:b3:cc:c5:77:a1)
    Type: 802.1Q Virtual LAN (0x8100)
▾ 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 0
    000. .... .... .... = Priority: Best Effort (default) (0)
    ...0 .... .... .... = DEI: Ineligible
    .... 0000 0000 0000 = ID: 0
    Type: IEC 61850/SV (Sampled Value Transmission (0x88ba)
▾ IEC61850 Sampled Values
    APPID: 0x0001
    Length: 102
    Reserved 1: 0x0000 (0)
    Reserved 2: 0x0000 (0)
  ▾ savPdu
      noASDU: 1
    ▸ seqASDU: 1 item

0000  08 00 27 f8 42 a7 a0 b3  cc c5 77 a1 81 00 00 00   ··'·B··· ··w·····
0010  88 ba 00 01 00 66 00 00  00 00 60 64 80 01 01 a2   ·····f·· ··`d····
0020  5f 30 5d 80 0c 46 52 45  41 2d 47 6f 53 56 2d 31   _0]··FRE A-GoSV-1
0030  20 82 02 00 08 83 04 00  00 00 01 85 01 00 87 40    ······· ·······@
0040  a2 18 80 16 00 00 00 a4  00 00 00 0c 00 01 5b fc   ········ ······[·
0050  f6 b0 00 3c 02 03 00 00  00 0d 00 00 00 6c 82 01   ···<···· ·····l··
0060  2e 9c 5a 20 e9 3e c2 54  d9 4b 5e 26 62 f7 9e 9c   .·Z ·>·T ·K^&b···
0070  c9 3f a8 5c 23 9e 56 e7  9a 94 b5 1d ca 90 ec d0   ·?·\#·V· ········
0080  17 25 b6 b4 04 22 77 ba  54 9f b6 9c 91 2b 2f bd   ·%···"w· T····+/·
0090  97 6d 8b 3a 71 9f e9 12  b5 54 e2 03 c6 c3 65 0e   ·m·:q··· ·T····e·
00a0  2d e8 70 1a 50 b4 47 56  96 af 0f 8a 18 a1 cb e0   -·p·P·GV ········
```

Figure 5.9: Packet Capture of Subscriber

The computational times required to generate R-SV messages with security extensions are calculated to assess timing performance. This is accomplished by sampling CPU times at the beginning and end of C programs. Table 5.2 shows the computational times for processing the SV frames, generating MAC values for various MAC algorithms, and encrypting the SV APDU at the publisher. It also shows the times taken to process the received SV frames, regenerate the MAC values for various MAC algorithms, and decrypt the SV APDU at the subscriber side. As seen in Table 5.2, the processing and communication delays for transmitting secure R-SV messages are significantly below the 3ms minimum. The simulation verifies the R-SV message structure provided in IEC 61850 standard part 9-2. In addition, the built security algorithm corresponds to the IEC 62531 security standard. Detailed data on the computational time is found in Appendix I and Appendix J. The end-to-end delay time is calculated with Wireshark data as provided in Appendix G. Average values are obtained from the values in the respective Appendix.

Table 5.2: Computational time for secure SV security algorithm

Security algorithm	Signature length (bytes)	Average computational time (ms)				
		EtM Publisher		EtM Subscriber		
		MAC generation time	Encryption time	MAC generation time	MAC comparison time	Decryption
AES 128	16	-	0.01044	-	-	0.001
HMAC SHA256	32	0.00226	-	0.008	0.001	-
End to End delay						
Normal	0.002088					
Worst case	0.001047					

Cybersecurity functions must not interrupt the existing protection functions of IEDs. Any delays or interruptions of the normal operation of the IED during the power system fault may damage or disrupt substation equipment. Therefore, the total time delay has to be measured to validate the performance of the security algorithm including delays in the merging unit, process bus Ethernet switch, IED, R-SV communication, GOOSE communication, and station bus Ethernet delays (Karnati, 2020).

Therefore, to confirm the security algorithm performance, the entire time delay needs to be measured. This includes the delays that occur in the merging unit, the process bus, the IED, the R-SV communication, the GOOSE communication, and the station bus Ethernet delays.

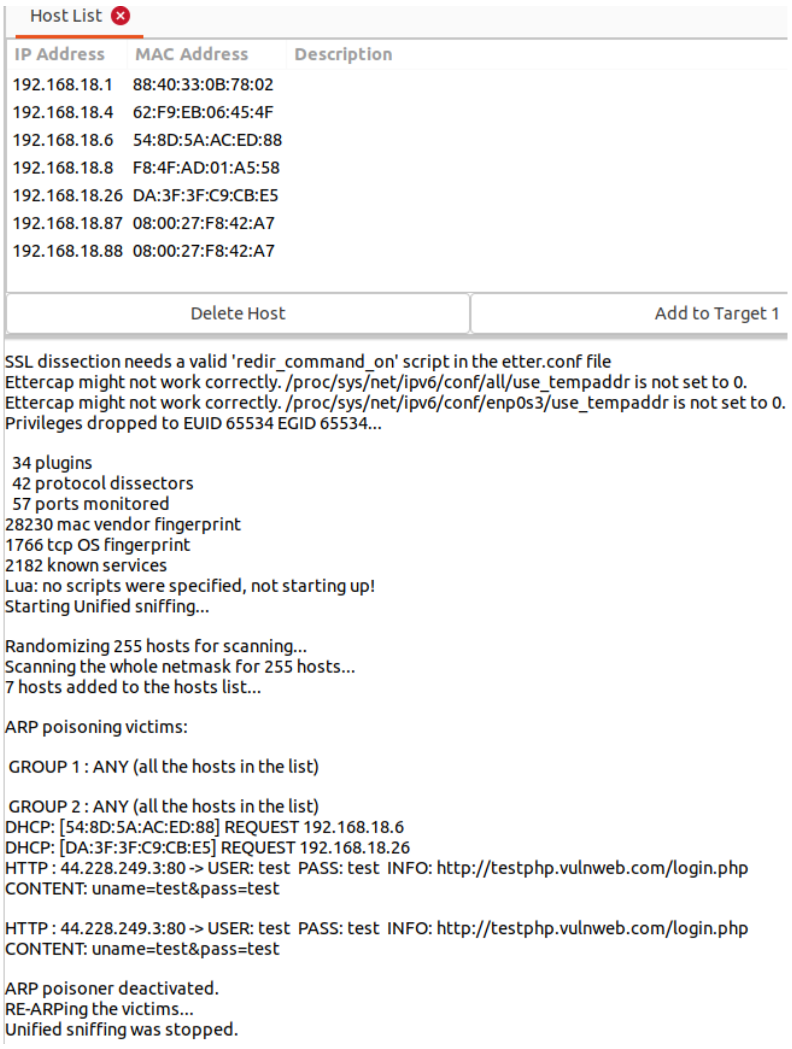
Using the S-GoSV library implementation, (Hussain, Farooq, et al., 2020) simulate a substation communication network to determine if a worst-case scenario will still be functional. Both processing and communication delays for transmitting encrypted GOOSE messages are within the 3ms threshold, as confirmed by the authors. Consequently, the proposed security method can be used effectively in real-time applications. Moreover, (Hussain, Farooq, et al., 2020) concur that the EtM security algorithm has greater benefits than the MtE and E&M algorithms. Based on these findings, future cybersecurity frameworks for the IEC 62351 standard can advocate encryption for IEC 61850 messages with confidence.

Overall, the adapted security scheme for SVMs in IEC 61850 provides a robust and comprehensive set of measures to ensure the confidentiality, integrity, and authenticity of the data. By incorporating encryption, timestamping, sequence numbers and HMACs the scheme provides a multi-layered defence against various types of attacks. This improved security scheme enables safe and secure communication between IEDs in the substation, helping to ensure the reliable and efficient operation of the power grid.

5.6 ARP MITM attacks using Ettercap and Wireshark

Due to the inclusion of vital equipment in substation-based communication and the transfer of data through an unsecured public network, a robust security mechanism is necessary to prevent cyberattacks. Multiple attacks, including MITM, replay, and DoS compromise substation-based communication based on the IEC 61850 framework. The developed secure R-SV algorithm can be utilised to examine the impact of various cyber vulnerabilities. The SV protocol exposes all data information in the communication network due to its features, including plain text messages and multicast at the data link layer. As a result, a hacker may discover vital information for cyberattacks. Exploiting the weaknesses of the processing subscriber is another technique to compromise the R-SV message and prevent the substation system from functioning normally. Attackers can launch cyberattacks if they get access to or reverse-engineer the security algorithm. ARP MITM attacks are also called ARP spoofing or ARP cache poisoning, the idea is to corrupt the ARP table of hosts using bogus ARP replies. The attacker needs to be on the same network as the hosts being attacked for this attack to work. This attack takes advantage of the lack of security mechanisms in ARP to validate the identities of ARP speakers. As such, the MITM attack is implemented on Ettercap, and Wireshark is used to capture and analyse the generated attacks of the packets.

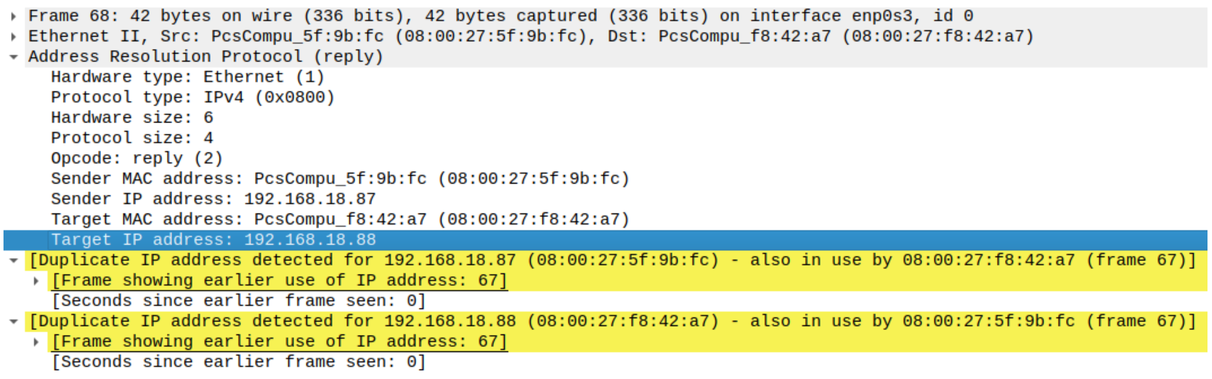
Data is transmitted from the publisher to the subscriber device. The publisher and subscriber devices are set as target 1 and target 2 respectively on Ettercap. An MITM attack is implemented to reroute the traffic as shown in Figure 5.10. As such, target 1 will assume that the attacker's MAC address is that of target 2, and traffic will be rerouted to the attacker as illustrated in Figure 5.11. The same applies when network traffic is transmitted via a router; the attacker will suggest that they are the target device and traffic must be rerouted via their MAC address. Once an adversary gains access to the process bus of the digital substation, they could monitor the R-SV packets and analyse the semantics of SV PDU. After finishing the analysis of SV streams, they could initiate the SV attacks such as injecting fault currents and voltages.



IP Address	MAC Address	Description
192.168.18.1	88:40:33:0B:78:02	
192.168.18.4	62:F9:EB:06:45:4F	
192.168.18.6	54:8D:5A:AC:ED:88	
192.168.18.8	F8:4F:AD:01:A5:58	
192.168.18.26	DA:3F:3F:C9:CB:E5	
192.168.18.87	08:00:27:F8:42:A7	
192.168.18.88	08:00:27:F8:42:A7	

```
SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Ettercap might not work correctly. /proc/sys/net/ipv6/conf/all/use_tempaddr is not set to 0.
Ettercap might not work correctly. /proc/sys/net/ipv6/conf/enp0s3/use_tempaddr is not set to 0.
Privileges dropped to EUID 65534 EGID 65534...

 34 plugins
 42 protocol dissectors
 57 ports monitored
28230 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!
Starting Unified sniffing...

Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
7 hosts added to the hosts list...

ARP poisoning victims:

 GROUP 1 : ANY (all the hosts in the list)

 GROUP 2 : ANY (all the hosts in the list)
DHCP: [54:8D:5A:AC:ED:88] REQUEST 192.168.18.6
DHCP: [DA:3F:3F:C9:CB:E5] REQUEST 192.168.18.26
HTTP : 44.228.249.3:80 -> USER: test  PASS: test  INFO: http://testphp.vulnweb.com/login.php
CONTENT: uname=test&pass=test

HTTP : 44.228.249.3:80 -> USER: test  PASS: test  INFO: http://testphp.vulnweb.com/login.php
CONTENT: uname=test&pass=test

ARP poisoner deactivated.
RE-ARPing the victims...
Unified sniffing was stopped.
```

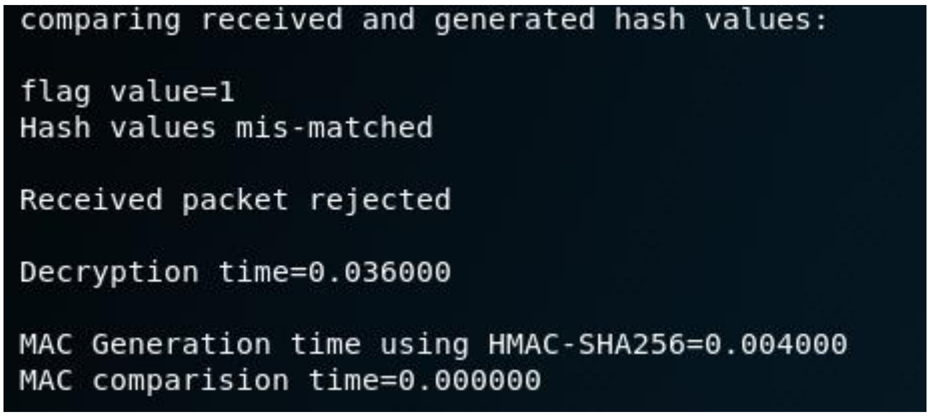
Figure 5.10: Ettercap capture of MITM attack



```
▸ Frame 68: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface enp0s3, id 0
▸ Ethernet II, Src: PcsCompu_5f:9b:fc (08:00:27:5f:9b:fc), Dst: PcsCompu_f8:42:a7 (08:00:27:f8:42:a7)
▾ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: PcsCompu_5f:9b:fc (08:00:27:5f:9b:fc)
    Sender IP address: 192.168.18.87
    Target MAC address: PcsCompu_f8:42:a7 (08:00:27:f8:42:a7)
    Target IP address: 192.168.18.88
▾ [Duplicate IP address detected for 192.168.18.87 (08:00:27:5f:9b:fc) - also in use by 08:00:27:f8:42:a7 (frame 67)]
  ▸ [Frame showing earlier use of IP address: 67]
    [Seconds since earlier frame seen: 0]
▾ [Duplicate IP address detected for 192.168.18.88 (08:00:27:f8:42:a7) - also in use by 08:00:27:5f:9b:fc (frame 67)]
  ▸ [Frame showing earlier use of IP address: 67]
    [Seconds since earlier frame seen: 0]
```

Figure 5.11: Wireshark capture of MITM attack

Furthermore, any change in the SV PDU during communication indicates a difference between the modified hash value (h) and the one recomputed at the receiver (h1). The SV PDU is tampered with and forwarded to the receiver. Tampering is discovered when hash values are recalculated. Figure 5.12 depicts the results of tamper detection when the SV PDU hash value is mismatched, and the received packet is discarded.



```
comparing received and generated hash values:

flag value=1
Hash values mis-matched

Received packet rejected

Decryption time=0.036000

MAC Generation time using HMAC-SHA256=0.004000
MAC comparision time=0.000000
```

Figure 5.12: Rejected SV PDU

In substation automation, mitigating cybersecurity attacks is a crucial requirement. As a result, the number of cyber-attacks on substations is on the rise, and it has emerged as a major threat that may cause substation damage. Encryption and authentication are essential security requirements that must be implemented to prevent cybersecurity attacks and provide robust security measures.

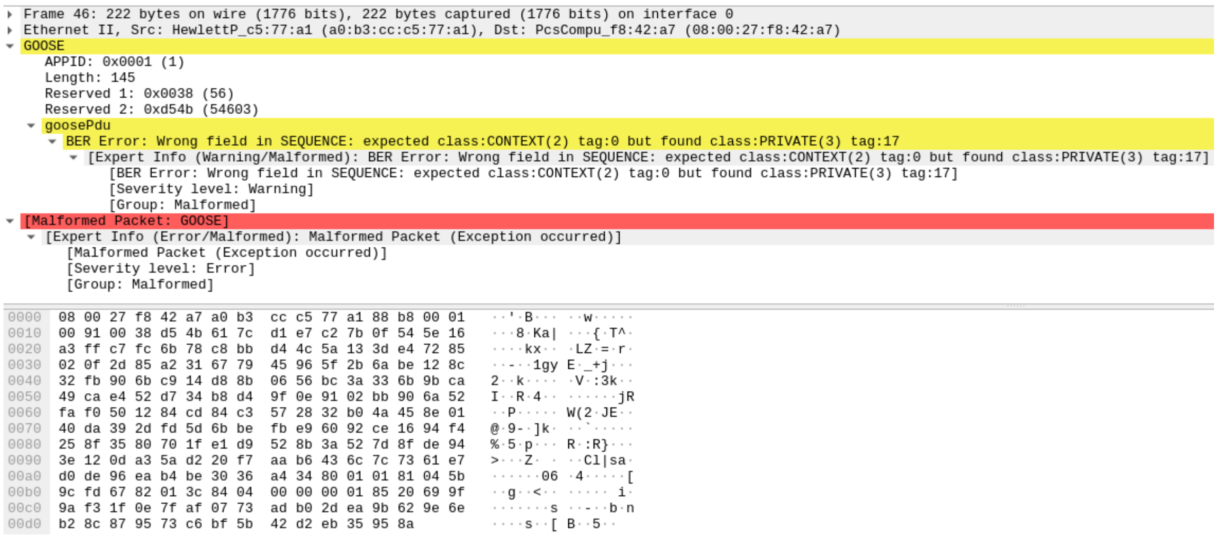
Authenticated encryption using the AES-HMAC algorithm provides robust security measures that ensure the confidentiality, integrity, and authenticity of data. The AES encryption algorithm is widely regarded as one of the most secure encryption algorithms, while the HMAC algorithm provides strong authentication and message integrity. The AES-HMAC algorithm is efficient in terms of processing time and computational resources, making it ideal for use in resource-constrained environments. The AES-HMAC algorithm is an efficient encryption method that can process large amounts of data quickly and securely. This is particularly important for the real-time transmission of R-SV messages, where delays in processing can have significant consequences. The use of authenticated encryption using the AES-HMAC algorithm is compliant with many industry standards such as IEC 62351.

5.7 Comparative Analysis

(Hussain, Farooq, et al., 2020) developed and implemented an EtM security algorithm on GOOSE PDU based on the modification of IEC 62351-6. However, a detailed analysis of the packet data on Wireshark was conducted and found that the GOOSE PDU format is not compliant with IEC 62351-6. Figure 5.13 shows the Wireshark capture of the non-compliant GOOSE PDU. If the format of GOOSE and R-SV are not adhered to as specified in IEC

61850 and IEC 62351, this may cause a potential low delay in delivering high-speed communication although the 3ms requirement will be met. Messages with a stringent time requirement must be delivered on time. Processing time delays at both ends, propagation time delays in communication links, and processing and queuing time delays in intermediate switches all contribute to transmission time. Non - compliance, on the other hand, may result in a lack of interoperability. The frame encryption and authentication procedures may no longer be compliant with the standard after modifications are made for implementation. Security may be significantly impacted by changing the encryption and authentication processes.

For successful implementation, the proposed additional security techniques on R-SVs must adhere to the above-mentioned timing requirements. The implementation to secure R-SV messages in the thesis is as defined in IEC 62351-6.



```
▸ Frame 46: 222 bytes on wire (1776 bits), 222 bytes captured (1776 bits) on interface 0
▸ Ethernet II, Src: HewlettP_c5:77:a1 (a0:b3:cc:c5:77:a1), Dst: PcsCompu_f8:42:a7 (08:00:27:f8:42:a7)
▾ GOOSE
    APPID: 0x0001 (1)
    Length: 145
    Reserved 1: 0x0038 (56)
    Reserved 2: 0xd54b (54603)
  ▾ goosePdu
    ▾ BER Error: Wrong field in SEQUENCE: expected class:CONTEXT(2) tag:0 but found class:PRIVATE(3) tag:17
      ▾ [Expert Info (Warning/Malformed): BER Error: Wrong field in SEQUENCE: expected class:CONTEXT(2) tag:0 but found class:PRIVATE(3) tag:17]
          [BER Error: Wrong field in SEQUENCE: expected class:CONTEXT(2) tag:0 but found class:PRIVATE(3) tag:17]
          [Severity level: Warning]
          [Group: Malformed]
▾ [Malformed Packet: GOOSE]
  ▾ [Expert Info (Error/Malformed): Malformed Packet (Exception occurred)]
      [Malformed Packet (Exception occurred)]
      [Severity level: Error]
      [Group: Malformed]

0000  08 00 27 f8 42 a7 a0 b3  cc c5 77 a1 88 b8 00 01   ··'·B···  ··w·····
0010  00 91 00 38 d5 4b 61 7c  d1 e7 c2 7b 0f 54 5e 16   ···8·Ka|  ···{·T^·
0020  a3 ff c7 fc 6b 78 c8 bb  d4 4c 5a 13 3d e4 72 85   ····kx··  ·LZ·=·r·
0030  02 0f 2d 85 a2 31 67 79  45 96 5f 2b 6a be 12 8c   ··-··1gy  E·_+j···
0040  32 fb 90 6b c9 14 d8 8b  06 56 bc 3a 33 6b 9b ca   2··k····  ·V·:3k··
0050  49 ca e4 52 d7 34 b8 d4  9f 0e 91 02 bb 90 6a 52   I··R·4··  ······jR
0060  fa f0 50 12 84 cd 84 c3  57 28 32 b0 4a 45 8e 01   ··P·····  W(2·JE··
0070  40 da 39 2d fd 5d 6b be  fb e9 60 92 ce 16 94 f4   @·9-·]k·  ··`·····
0080  25 8f 35 80 70 1f e1 d9  52 8b 3a 52 7d 8f de 94   %·5·p···  R·:R}···
0090  3e 12 0d a3 5a d2 20 f7  aa b6 43 6c 7c 73 61 e7   >···Z· ·  ··Cl|sa·
00a0  d0 de 96 ea b4 be 30 36  a4 34 80 01 01 81 04 5b   ······06  ·4·····[
00b0  9c fd 67 82 01 3c 84 04  00 00 00 01 85 20 69 9f   ··g··<··  ····· i·
00c0  9a f3 1f 0e 7f af 07 73  ad b0 2d ea 9b 62 9e 6e   ·······s  ··-··b·n
00d0  b2 8c 87 95 73 c6 bf 5b  42 d2 eb 35 95 8a         ····s··[  B··5··
```

Figure 5.13: Wireshark capture of GOOSE PDU with error

If the IEC 61850 GOOSE frame structure does not adhere to the standard requirements, various issues may arise. These disadvantages can affect the overall performance, interoperability, and reliability of the substation automation system. Below are some specific drawbacks that can occur.

1. Compatibility Issues: Adhering to the IEC 61850 standard guarantees compatibility between devices from different vendors, enabling them to work together effectively. However, if the GOOSE frame structure does not meet the standard, it can lead to incompatibility issues among devices produced by different manufacturers. As a consequence, this can cause communication difficulties, data loss, or incorrect data

interpretation, ultimately obstructing the smooth integration of devices in the substation.

2. Communication Errors: The GOOSE communication system relies on a defined frame structure outlined in the standard. Any deviations from this structure can lead to problems in communication. Incompatible frame structures can cause incorrect interpretation of data or even the loss of crucial information during transmission. As a consequence, this poses a risk to the security and efficiency of operations involving IEDs and other time-sensitive services within the substation.
3. Reduced Performance: Adhering to the standard guarantees efficient and optimal communication among IEDs. The purpose of the GOOSE frame structure is to minimize network bandwidth usage while facilitating rapid data exchange. However, frame structures that do not meet the standard can introduce unnecessary overhead or inadequate data representation, leading to increased network traffic, longer delays, and a decline in overall system performance.

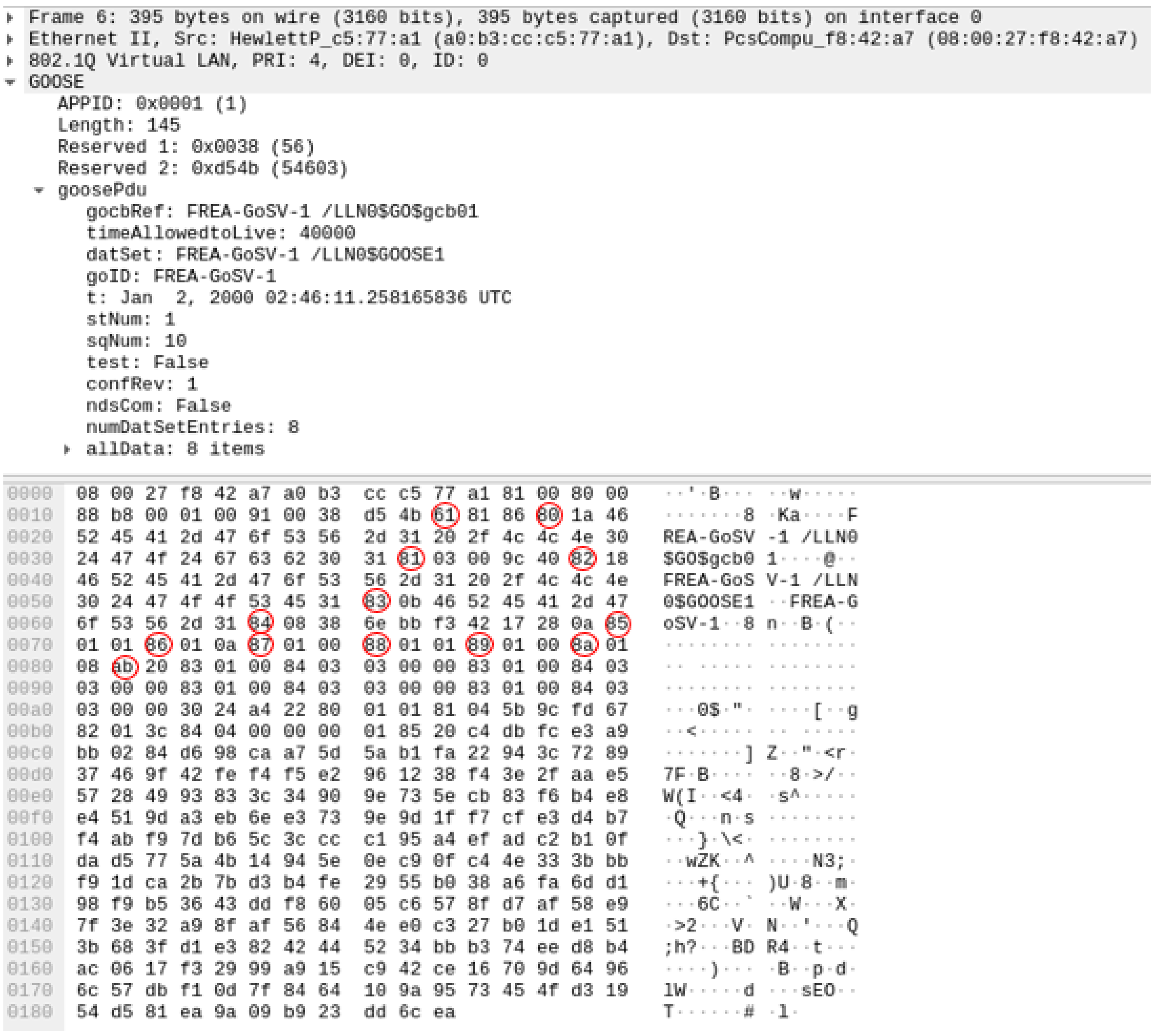
The format structure outlined in IEC 62351-6:2020 is essential for establishing secure and dependable communication. However, if the format structure does not adhere to the standard, it can result in various drawbacks, such as:

1. Security Vulnerabilities: The IEC 62351-6:2020 standard outlines a format structure that encompasses authentication, encryption, and integrity verification methods. Failure to adhere to this structure or implement it accurately can lead to security weaknesses. Unauthorized individuals may exploit these weaknesses to gain entry without permission, tamper with data, or carry out harmful actions. Failing to meet the standard raises the likelihood of security breaches and undermines the overall security of power and energy systems.
2. Interoperability Issues: Adhering to IEC 62351-6:2020 guarantees efficient and secure communication among various components and systems in the power and energy infrastructure. However, if the format structure fails to comply with the standard, it can lead to problems with interoperability. This means that components produced by different manufacturers may struggle to interpret or handle communication data accurately, resulting in communication breakdowns, data errors, or system instability. Non-compliance hampers the integration and seamless functioning of diverse systems and components.
3. Reduced Resiliency: The standard's defined format structure guarantees the trustworthiness and consistency of communication between control systems and their associated parts. Failure to adhere to this structure can result in errors or discrepancies

during the transmission and reception of data. These errors have the potential to cause misunderstandings in commands, incorrect system reactions, or even complete system breakdowns. Format structures that do not comply with the standard can jeopardize the overall reliability of power and energy systems, impacting their efficiency and accessibility. Non-compliant systems may display abnormal behaviour, including variations in message formatting, timing, or protocol implementation.

To address these drawbacks, it is crucial to guarantee adherence to the format structure outlined in IEC 62351-6:2020. By following this standard, organisations can bolster the security, interoperability, reliability, and regulatory adherence of their power and energy systems. This, in turn, promotes safer and more efficient operations within this vital industry.

As depicted in Figure 5.13, the presence of an error in the packet indicates the existence of significant issues, such as malformed packets. These problems involve a violation of the specification of the GOOSE protocol, which includes invalid field values or illegal lengths. In Figure 5.14, the Wireshark capture illustrates a compliant GOOSE PDU. The code presented by (Hussain, Farooq, et al., 2020) has been adjusted to align with the standards of IEC 61850 and IEC 62351. Modifications may be necessary for frame encryption and authentication due to non-compliance with the specified format. Format non-compliance refers to situations where the data frame format used in the system does not meet the requirements outlined in the standard.



```
▸ Frame 6: 395 bytes on wire (3160 bits), 395 bytes captured (3160 bits) on interface 0
▸ Ethernet II, Src: HewlettP_c5:77:a1 (a0:b3:cc:c5:77:a1), Dst: PcsCompu_f8:42:a7 (08:00:27:f8:42:a7)
▸ 802.1Q Virtual LAN, PRI: 4, DEI: 0, ID: 0
▾ GOOSE
    APPID: 0x0001 (1)
    Length: 145
    Reserved 1: 0x0038 (56)
    Reserved 2: 0xd54b (54603)
  ▾ goosePdu
      gocbRef: FREA-GoSV-1 /LLN0$GO$gcb01
      timeAllowedtoLive: 40000
      datSet: FREA-GoSV-1 /LLN0$GOOSE1
      goID: FREA-GoSV-1
      t: Jan  2, 2000 02:46:11.258165836 UTC
      stNum: 1
      sqNum: 10
      test: False
      confRev: 1
      ndsCom: False
      numDatSetEntries: 8
    ▸ allData: 8 items

0000  08 00 27 f8 42 a7 a0 b3  cc c5 77 a1 81 00 80 00   ··'·B···  ··w·····
0010  88 b8 00 01 00 91 00 38  d5 4b 61 81 86 80 1a 46   ·······8  ·Ka····F
0020  52 45 41 2d 47 6f 53 56  2d 31 20 2f 4c 4c 4e 30   REA-GoSV -1 /LLN0
0030  24 47 4f 24 67 63 62 30  31 81 03 00 9c 40 82 18   $GO$gcb0 1····@··
0040  46 52 45 41 2d 47 6f 53  56 2d 31 20 2f 4c 4c 4e   FREA-GoS V-1 /LLN
0050  30 24 47 4f 4f 53 45 31  83 0b 46 52 45 41 2d 47   0$GOOSE1 ··FREA-G
0060  6f 53 56 2d 31 84 08 38  6e bb f3 42 17 28 0a 85   oSV-1··8 n··B·(··
0070  01 01 86 01 0a 87 01 00  88 01 01 89 01 00 8a 01   ········ ········
0080  08 ab 20 83 01 00 84 03  03 00 00 83 01 00 84 03   ·· ····· ········
0090  03 00 00 83 01 00 84 03  03 00 00 83 01 00 84 03   ········ ········
00a0  03 00 00 30 24 a4 22 80  01 01 81 04 5b 9c fd 67   ···0$·"· ····[··g
00b0  82 01 3c 84 04 00 00 00  01 85 20 c4 db fc e3 a9   ··<····· ·· ·····
00c0  bb 02 84 d6 98 ca a7 5d  5a b1 fa 22 94 3c 72 89   ·······] Z··"·<r·
00d0  37 46 9f 42 fe f4 f5 e2  96 12 38 f4 3e 2f aa e5   7F·B···· ··8·>/··
00e0  57 28 49 93 83 3c 34 90  9e 73 5e cb 83 f6 b4 e8   W(I··<4· ·s^·····
00f0  e4 51 9d a3 eb 6e e3 73  9e 9d 1f f7 cf e3 d4 b7   ·Q···n·s ········
0100  f4 ab f9 7d b6 5c 3c cc  c1 95 a4 ef ad c2 b1 0f   ···}·\<· ········
0110  da d5 77 5a 4b 14 94 5e  0e c9 0f c4 4e 33 3b bb   ··wZK··^ ····N3;·
0120  f9 1d ca 2b 7b d3 b4 fe  29 55 b0 38 a6 fa 6d d1   ···+{··· )U·8··m·
0130  98 f9 b5 36 43 dd f8 60  05 c6 57 8f d7 af 58 e9   ···6C··` ··W···X·
0140  7f 3e 32 a9 8f af 56 84  4e e0 c3 27 b0 1d e1 51   ·>2···V· N··'···Q
0150  3b 68 3f d1 e3 82 42 44  52 34 bb b3 74 ee d8 b4   ;h?···BD R4··t···
0160  ac 06 17 f3 29 99 a9 15  c9 42 ce 16 70 9d 64 96   ····)··· ·B··p·d·
0170  6c 57 db f1 0d 7f 84 64  10 9a 95 73 45 4f d3 19   lW·····d ···sEO··
0180  54 d5 81 ea 9a 09 b9 23  dd 6c ea                  T······# ·l·
```

Figure 5.14 Wireshark capture of correct GOOSE PDU

It should be emphasised that Abstract Syntax Notation One (ASN.1) Basic Encoding Rules (BER) are utilised for the decoding of GOOSE and SV protocols. The BER transfer syntax operates on the principle of a triplet format known as Tag, Length, Value (TLV). As demonstrated in Figure 5.14, the data sequence follows the BER structure, as shown in Figure 5.15, specifically for GOOSE. The GOOSE PDU is encoded using the TLV format, with the starting tag being 0x61. Subsequently, the Tag is followed by the Length, which indicates the overall length of the GOOSE PDU. The Value section of the GOOSE PDU contains a sequence of data.

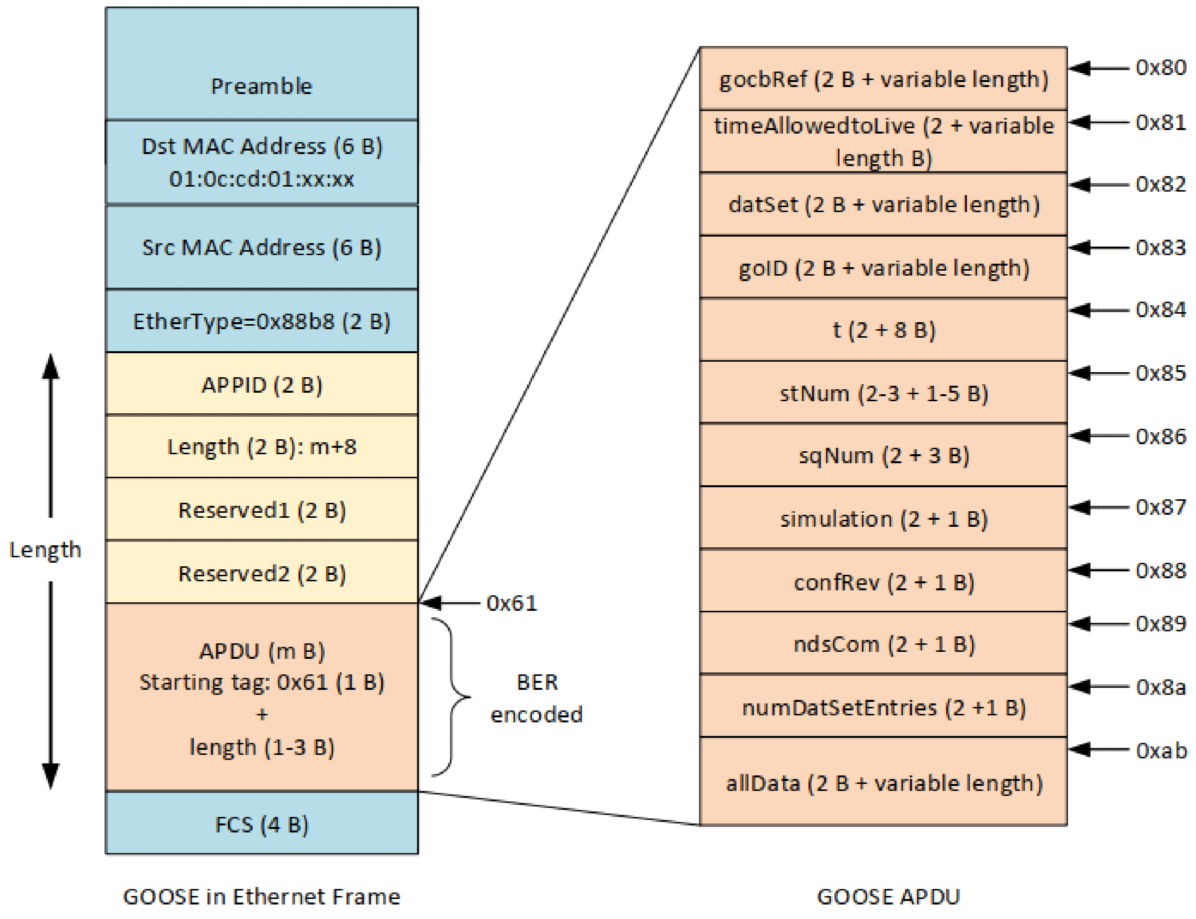


Figure 5.15 IEC 61850-8-1 GOOSE PDU structure (ASN.1 Encoding)

(Rodriguez et al., 2021) analysed and evaluated different security algorithms as referred to in Chapter 2, however, the authors have recommended the use of the AES Galois Counter Mode (AES-GCM) algorithm. The authors have proven that even in a worst-case scenario the algorithm is extremely efficient and achieves both data throughput and low latency when implemented in hardware. The presented solution is fully IEC 62351-6 compliant. (Suhail Hussain et al., 2023) recently discovered that processing time delays at both the publisher and subscriber for different MAC algorithms (AES-HMAC-128/256, AES-GCM-128/256, etc.) are less than the 0.2ms limit. They validated that the overall end-to-end delays, including processing and communication time delays, for various MAC algorithms are less than the IEC 61850 standards' 3ms restriction. When choosing between these algorithms, it is important to consider the specific security requirements and potential attack scenarios of the system being secured.

Both AES-HMAC and AES-GCM algorithms provide strong security measures that ensure the confidentiality, integrity, and authenticity of data. As computing capabilities increase, it's generally recommended to use longer key lengths for increased security. To ensure that the

3ms timing requirements are met, opt for encryption algorithms that strike a balance between security and computational efficiency, optimise the code and hardware for better performance, and optimise the overall network infrastructure for low-latency communication to minimise congestion. Although loner key lengths provide higher security, always assess the security requirements, and choose minimum key lengths to help reduce computational overheads.

However, AES-GCM is generally considered more secure due to its use of the GCM authentication mechanism, which provides better protection against potential attacks. Both AES-HMAC and AES-GCM algorithms are efficient in terms of processing time and computational resources. However, AES-GCM is generally considered more efficient due to its parallel processing capabilities, which allow for faster encryption and decryption of large amounts of data. As with any encryption algorithm, the security of AES-GCM depends heavily on the proper management of cryptographic keys. Weaknesses in key generation, storage, or distribution could compromise the security of the system. To ensure that key management is not compromised, use a secure number generator and ensure that keys are long enough to resist brute-force attacks. Use secure key storage mechanisms and ensure the use of secure channels for key distribution. Avoid transmitting keys over insecure networks or channels.

Existing methods and techniques for securing R-SV messages in IEC 61850 communication include Digital Signature (DS), Message Authentication Code (MAC), and Transport Layer Security (TLS). DS and MAC provide authentication and integrity protection but do not provide encryption. TLS provides both encryption and authentication but can be more complex and costly to implement compared to authenticated encryption. In comparison to these existing methods and techniques, authenticated encryption offers a good balance of security and efficiency. It provides both encryption and authentication in a single operation, reducing processing overhead and improving performance. Additionally, authenticated encryption offers strong confidentiality and integrity protection, ensuring that the data transmitted between two parties is secure and authentic. However, the key management system needs to be secure, and it is vulnerable to side-channel attacks, which are also limitations of other cryptographic techniques.

5.8 Conclusion

The IEC 61850 standard is gaining more attention as it is positioned to become the next standard for power system communication. A significant portion of the research focuses on adapting IEC 61850 information models and message structures to new smart grid devices. Performance evaluations need the presence of a tool that can generate and distribute GOOSE and SV messages with specific parameters. This chapter describes in detail the implementation processes that led to the development of the secure R-SV functions. The programming for the encapsulation of different layers of the R-SV frame in C programming under the Linux system is also documented. The implemented simulation demonstrates that messages published by the R-SV source code strictly adhere to the IEC 61850 format, as identified by the Wireshark network sniffer software tool, which correctly decoded all fields of the generated custom GoSV frames. SV streams are utilized for the purpose of real-time monitoring and control within a networked system. The quantity of SV streams that can be disseminated across the system's network is contingent upon several factors. These factors include the capacity of the communication network, which necessitates the utilization of high-performance IEEE 802.1Q-compliant managed Ethernet switches on the process bus, as well as the processing capabilities of the devices, ensuring compliance with the IEC 61850 standard. In addition to considerations regarding bandwidth limitations, it is imperative to carefully manage the configuration of the Process Bus Local Area Network (LAN) segment and to meticulously select Layer 2 multicast addresses. IEEE 802.1Q facilitates both the processing of messages with priority-based scheduling policies and the segmentation of the process bus to enable efficient processing of SV streams.

The EtM algorithm is proposed for maintaining message confidentiality and integrity. IEC 62351 is employed to implement the security requirements for R-SV messages. For privacy, the EtM algorithm is implemented with AES-128 encryption. MAC algorithms are employed to authenticate messages. Simulation results indicate that the EtM algorithm can successfully be used for R-SV messages while meeting stringent 3 ms latency criteria. The findings indicate that the proposed MAC and AES algorithms can be implemented in R-SV communications without any challenges. Based on these findings, future IEC 62351 security standards can confidently advocate encryption for R-SV messages.

6. CHAPTER SIX: CONCLUSION AND FUTURE WORK

6.1 Introduction

It is difficult to secure today's power systems since they frequently employ communication protocols with minimal or no security mechanisms yet are implemented for bandwidth and efficiency. Additionally, many grids haven't received security upgrades post-commissioning. To address vulnerable power grids, IEC Technical Committee 57 began research on ways to make power grids secure in the early 2000s. WG15 was established to assess the requirements from a technical standpoint and determine a method for implementation. As such, the IEC 61850 standard represents a significant step forward for both standardisation and ICS security for digital substations. With its extensive implementation, utilities and operators can now efficiently commission, collaborate, and maintain new equipment (Carullo, 2020). However, although being efficient, its communication protocols, GOOSE and SV, have security vulnerabilities. While numerous scholars from all over the world have proposed solutions to the issue, IEC has already established a standard approach to deal with such flaws.

Electricity generation, substation, and power grid operations are being compelled to strengthen OT and IoT security methods to increase the resilience of their systems in response to rising cyber-attacks, management concerns, and governmental policies. Innovative solutions that improve OT and IoT visibility, cybersecurity, and availability are critical components that must be implemented. For event visibility, OT and IoT systems are required at the grid or substation level. The reliability and safety of the power system may be negatively impacted by networking issues. The ability to react quickly to threats and abnormalities is essential, but early detection of problems necessitates real-time visibility over connections, communications, and other factors. Unfortunately, many power systems lack these capabilities. Operational reliability can be significantly affected by security weaknesses in operations and technology. More emphasis should be placed on best practices and technology to improve the reliability and security of the electrical system. By monitoring network traffic for security attacks and suspicious activity and further delivering enhanced security detection, Nozomi Networks offers a solution to boost OT and IoT visibility. The solution from Nozomi Networks enhances visibility, resilience, and cybersecurity. (Carullo, 2020)

To ensure the cross-vendor interoperability that has made IEC 61850 effective, researchers have emphasised the significance of establishing a single standard in substation automation systems. To achieve integrity and confidentiality, IEC 61850 communication messages must now be deployed with security mechanisms. IEC 62351-6 specifies a method for securing IEC 61850 protocols by incorporating a security extension section into the frames. Security threats in IEC 61850 communication messages have been addressed and vendors are standardising implementation to protect the IEC 61850 protocols. Enhancing the IEC 62351 security standard will ensure the security of the electricity grid.

The rising number of cyber-physical attacks on the power system demonstrates the necessity to improve the security mechanisms of existing industrial communication protocols. Although GMAC and HMAC are suggested by IEC62351- 6:2020 as cybersecurity mitigation to verify SV integrity, real-time applications, and performance evaluations for using the MAC algorithms have not been fully implemented. Compromised security keys between publisher and subscriber may disclose additional security vulnerabilities and cyber threats. This thesis recommended the implementation of a secure R-SV framework to address the abovementioned issue by evaluating the developed algorithm in a LAN environment. The development of the security scheme for R-SV messages was a comprehensive process that involved careful consideration of the various security threats that could affect the integrity, confidentiality, and availability of the sampled value data. The scheme was designed to provide a robust and effective solution to these threats, while also ensuring that the requirements of the standard were met. The performance of the proposed R-SV message has been analysed and validated with AES and HMAC algorithms. The results of the secure R-SV framework meet the performance requirements of IEC 61850. This can be applied to test benches of IEDs for further implementation in a real environment. The cryptographic strength of encryption and authentication techniques determines the security of a system. We investigated the combination of an encryption and authentication algorithm, taking the algorithm's cryptographic strength and performance into account. In our studies, we discovered that AES-128 is a more efficient encryption method with higher performance, whereas HMAC-SHA256 is a more efficient authentication algorithm. However, in certain environments, there are several concerns and potential drawbacks to employing AES-128. The main worry with AES-128 is its relatively low-key length, which may render it more vulnerable to brute-force attacks or new attacks over time (due to algorithm ageing). AES-128 deployment in a substation environment necessitates adequate design and management. Inadequate configuration settings, such as poor key management or ineffective modes of operation, could compromise encryption security.

Digital substations must be structurally organised to ensure that the sampled value method is always operational. The performance of protection systems is therefore of the utmost importance to maintain the reliable operation of the power grid. As IEC 61850 becomes more prevalent in power systems, sampled values-based fault detection and isolation solutions will inevitably become the industry standard. Therefore, they must be thoroughly verified and evaluated to be as reliable as traditional protection structures as secure communications will be crucial for the next generation of technology within substation automation. Section 6.2 describes the problems that were solved in this thesis. The thesis deliverables are discussed in Section 6.3. The algorithm developed is discussed in Section 6.4, and Section 6.5 proposes future work. Section 6.6 discusses how the work done in this study has been applied. Section 6.7 provides the conclusion to this chapter.

6.2 Problems Solved in this Thesis

The problems solved can be categorised into two sections which are:

- Design-based
- Implementation-based

6.2.1 Design-based Problems

Sub-problem 1: Overview and analysis of IEC 61850 protocols in particular GOOSE and Sampled Values.
Sub-problem 2: Overview of IEC 62351 cyber security implementation for smart grids and in-depth analysis of IEC 62351-6.
Sub-problem 3: Critical analysis of time requirements of IEC 61850 protocols.
Sub-problem 4: Overview and critical analysis of encryption and authentication techniques.
Sub-problem 5: Design and development of a secure R-SV message security algorithm.
Sub-problem 6: Critical analysis of the developed secure R-SV algorithm with existing security algorithms.

6.2.2 Implementation-based Problems

Sub-problem 1: Simulation of the authenticated encryption algorithm using Kali-Linux virtual machine and obtain results in Wireshark software for comparative analysis.
Sub-problem 2: Simulation of an MITM attack via Ettercap software and analysis of the network traffic.
Sub-problem 3: Simulation and comparative analysis of other available security algorithms, providing improvisations and amendments if required.

6.3 Thesis Deliverables

The following deliverables have been achieved through the work done in this thesis.

6.3.1 Literature Review

To obtain a thorough understanding of IEC 61850 and IEC 62351, a detailed literature review was conducted. The GOOSE and Sampled Value protocols, as well as security algorithms, received special attention. The development of various security techniques for a secure power system is discussed, demonstrating that this is an active research subject. According to this evaluation, security algorithms are expected to meet cyber security and timing requirements for GOOSE and Sampled Values communications. This literature analysis inspired the idea to create a security method based on authenticated encryption.

6.3.2 Critical Analysis of IEC 61850

The IEC 61850 standard considers substation automation network communication requirements. As the central smart grid communication protocol, IEC 61850 provides an integrated solution in the power system for communication between intelligent devices, ensuring interoperability and long-term stability while including a higher form of standardisation. IEC 61850 is the optimal communication standard for substation automation based on protocols and standards. The usage of the GOOSE message at a substation is crucial for power system protection. The GOOSE and SV operations use high-speed switched Ethernet data frames with no middle-layer processing. However, IEC 61850 lacks any security-related elements, and cyber-security threats in the substation environment remain a problem.

6.3.3 Critical Analysis of IEC 62351

To address this risk in power systems and further enhance cyber security measures for GOOSE and SV communication, IEC 62351 has been implemented. It is critical to protect SAS communication from cyber-security threats. Implementing IEC 61850 and IEC 62351 standards necessitates a thorough understanding of data networking, software modelling, system simulation, and testing procedures. Networking applications are becoming increasingly popular for secure communication. It is critical to provide a system for verifying data security transmission across an unstable and unsecure medium. The need for secure data transfer has prompted the creation of cryptographic standards and encryption techniques. The HMAC is a recommended authentication standard with strong security characteristics. Further research uses RSA algorithms for their advantages in encryption and authentication to give better security, but they have the downside of being time-consuming and requiring more computing power. To achieve confidentiality and integrity, the study uses an authenticated encryption technique. The security requirements for GOOSE/SV communications are implemented using IEC 62351. The authenticated encryption procedure is applied with AES-128 encryption for privacy. Messages are authenticated using MAC techniques.

6.3.4 Design and Development of an Authenticated Encryption Algorithm

A review of the most recent GOOSE and SV message security options was done. To assure compliance with IEC 61850-9-2 and IEC 62351-6, the presented authenticated encryption method code was designed, and put into practice, and the results were examined. Stringent performance specifications apply to GOOSE and SV communications, which are essential for secure operation. Compared to non-secure protocols, Secure SV offers a high level of security. Data is shielded from interception, modification, and tampering when secure SV messages are delivered with extra security measures like authentication and encryption. Secure SV can therefore be used in critical systems where data integrity and confidentiality are crucial. The designed security algorithm meets data and sampling speed requirements for performance and security. The algorithm developed publishes R-SV communication and is IEC 61850 compliant, while the applied authentication and integrity levels are IEC 62351-6 compliant. To allow authentication and authorisation, critical security elements must be enabled. For message confidentiality and integrity, the EtM method is recommended, together with AES-128 encryption for privacy and MAC techniques for message authentication. The simulation results show that the EtM technique can be employed for R-SV messages while meeting the strict 3 ms time limitation. The findings imply that future IEC 62351 security standards can confidently advocate for SV communication encryption.

6.4 Future Work

- Performance evaluation of multiple R-SV streams: This involves evaluating the sampled data accuracy during transmission and reception. In order to ensure that the time alignment complies with the required requirements, it entails assessing the synchronisation of sampled data.

- Investigate efficient key management among substation devices: For substation devices which comprise different types of IEDs, effective key management is crucial to securing communication channels, protecting sensitive data, and maintaining the power grid's overall cybersecurity.
- Smart grid security software testing and security validation: As with any new technology, smart grid security is important to preventing cyberattacks and ensuring a secure, uninterrupted supply of electricity. Evaluating the encryption mechanisms used to secure data during transmission and storage. Ensure that data integrity is maintained and that sensitive information is properly encrypted.
- Real-time simulation and performance analysis of a cyber-power system with different security algorithms: Establish a cyber-power system model by implementing a real-time simulation environment. Incorporate the chosen security algorithms into the simulation environment that runs in real-time. Assess how well they function in a dynamic environment by taking into account variables like response speed, resource utilisation, and flexibility in response to changing circumstances.

6.5 Application of the results from this thesis

The research, methods, and algorithms that have resulted from the work done in this thesis can be utilised for the following purposes:

- Used in smart grid systems for smart metering and energy automation.
- Used in substation automation systems.
- The security algorithms can be used in practical applications at Cape Peninsula University of Technology for research purposes.

6.6 Conclusion

This chapter presents a summary of the work conducted in this thesis. Software algorithms in the form of authenticated encryption have been developed to achieve the aims and objectives of the work done in this thesis.

REFERENCES

(Nozomi), Ite, W.H. & Pe, P.A. 2019. Improving ICS Cyber Security for Substations and Power Grids Real-time ICS Threat Detection and Operational Visibility Use Cases. , (August).

Abdolkhalig, A. 2014. Dynamic Phasor Estimation in Electrical Power Systems Based on IEC61850 Process-Bus. , (1–131). https://digital.library.adelaide.edu.au/dspace/bitstream/2440/92054/3/02whole.pdf.

Adewole, A.C. & Tzoneva, R. 2014a. *Impact of IEC 61850-9-2 standard-based process bus on the operating performance of protection IEDS: Comparative study*. IFAC. http://dx.doi.org/10.3182/20140824-6-ZA-1003.00598.

Adewole, A.C. & Tzoneva, R. 2014b. Impact of IEC 61850-9-2 standard-based process bus on the operating performance of protection IEDS: Comparative study. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 19(January): 2245–2252.

Alajbegović, H., Zečić, D. & Jamak, H. 2006. Digital Signature Algorithm. *10th International Research/Expert Conference*, (September).

Ali, N.H., Ali, B.B.M., Basir, O., Othman, M.L. & Hashim, F.B. 2016. WLAN oriented optimization of process bus in IEC 61850-based substation communication network. *Proceedings of 2015 IEEE International Renewable and Sustainable Energy Conference, IRSEC 2015*, (June 2017).

Apostolov, A. 2010. IEC 61850 9-2 process bus applications and benefits. *IET Conference Publications*, 2010(558 CP).

Apostolov, A. & Vandiver, B. 2011. IEC 61850 GOOSE applications to distribution protection schemes. *2011 64th Annual Conference for Protective Relay Engineers*, (December): 178–184.

Bertocco, M., Ferraris, F., Offelli, C., Parvis, M., Blanco, J.R., Ferrero, F.J., Valledor, M., Campo, J.C., Haizad, M., Ibrahim, R., Adnan, A., Chung, T.D., Hassan, S.M., Sharma, D.P., Samuel, K., Ramoutar, K., Lowe, T., David, I., Kalaitzakis, K., Koutroulis, E., Vlachos, V., Moreno, C., González, A., Olazagoitia, J.L., Vinolas, J., Wali, S. & Areeb, M. 1998. Real Time Communication with Client / Server Architecture Using Secure Shell Protocol. , 47(1–5): 1–6.

Bhamare, Y. Utilization of IEC 61850 GOOSE messaging in protection applications in distribution network.

Bhanot, R. & Hans, R. 2015. A review and comparative analysis of various encryption algorithms. *International Journal of Security and its Applications*, 9(4): 289–306.

Bogdanov, A., Khovratovich, D. & Rechberger, C. 2011a. *Biclique cryptanalysis of the full AES*.

Bogdanov, A., Khovratovich, D. & Rechberger, C. 2011b. *Biclique cryptanalysis of the full AES*.

Chen, F. & Yuan, J. 2012. Enhanced key derivation function of HMAC-SHA-256 algorithmin LTE network. *Proceedings - 2012 4th International Conference on Multimedia and Security, MINES 2012*, 3: 15–18.

Cleveland, F.M. 2012. IEC TC 57 WG15: IEC 62351 Security Standards for the ower System Information Infrastructure. *International Electrotechnical Commission*, 14.

Commission, I.E. 2017. IEC TR 61850-7-500. *IEC Technical Report*, 1.0: 1–10.

Committee, S., Power, I. & Society, E. 2017. *IEEE Recommended Practice for Implementing an IEC 61850-Based Substation Communications , Protection , Monitoring , and Control System IEEE Power and Energy Society IEEE Recommended Practice for Implementing an IEC 61850-Based Substation Communications , .*

Dolezilek, D., Gammel, D. & Fernandes, W. 2020. Cybersecurity based on IEC 62351 and IEC 62443 for IEC 61850 systems. *IET Conference Publications*, 2020(CP771).

Dondossola, G. & Terruggia, R. 2015. *Cyber Physical Systems Approach to Smart Electric Power Grid*. http://www.scopus.com/inward/record.url?eid=2-s2.0-84921709913&partnerID=tZOtx3y1.

Elbez, G., Keller, H.B. & Hagenmeyer, V. 2018. A Cost-efficient Software Testbed for Cyber-Physical Security in IEC 61850-based Substations. *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids, SmartGridComm 2018*: 1–6.

Elbez, G., Keller, H.B. & Hagenmeyer, V. 2019. Authentication of GOOSE Messages under Timing Constraints in IEC 61850 Substations. : 137–143.

Emmanuel, L. 2014. *VIRTUALIZATION OF A SENSOR NODE TO ENABLE THE SIMULATION OF IEC 61850-BASED SAMPLED VALUE MESSAGES*.

Engler, F., Kruimer, B., Kern, T.L., Schimmel, G., Andersson, L. & Schwarz, K. 2004. IEC 61850 based digital communication as interface to the primary equipment. : 1–8.

Farooq;, S.M., Hussain, S.M.S. & Ustun, T.S. 2019. S-GoSV: Framework for Generating Secure IEC 61850 GOOSE and Sample Value Messages.

Farooq, S.M., Hussain, S.M.S., Kiran, S. & Ustun, T.S. 2018. Certificate based authentication mechanism for PMU communication networks based on IEC 61850-90-5. *Electronics (Switzerland)*, 7(12).

Farooq, S.M., Hussain, S.M.S. & Ustun, T.S. 2019. Performance evaluation and analysis of IEC 62351-6 probabilistic signature scheme for securing GOOSE messages. *IEEE Access*, 7(March): 32343–32351.

Fernandes, C., Borkar, S. & Gohil, J. 2014. Testing of Goose Protocol of IEC61850 Standard in Protection IED. *International Journal of Computer Applications*, 93(16): 30–35.

Firouzi, S.R., Vanfretti, L., Ruiz-Alvarez, A., Hooshyar, H. & Mahmood, F. 2017. 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(March): 255–267.

Francis, N. & Monoth, T. 2018. An Analysis of Hybrid Cryptographic Approaches for Information Security. *International Journal of Applied Engineering Research*, 13(3): 124–127. http://www.ripublication.com.

Gadelha Da Silveira, M. & Franco, P.H. 2019. IEC 61850 Network Cybersecurity: Mitigating GOOSE Message Vulnerabilities.

Gonzalez-Redondo, M.J., Moreno-Munoz, A., Pallares-Lopez, V., Real-Calvo, R.J., Lopez, M.A.O. & Moreno-Garcia, I.M. 2013. IEC 61850 GOOSE transfer time measurement in development stage. *IEEE International Symposium on Industrial Electronics*.

Groat, J., Vandiver, G.S.A.B. & Vasudevan, B. 2023. Communication bandwidth considerations for digital substation applications. *2023 76th Annual Conference for Protective Relay Engineers, CFPR 2023*: 1–12.

Gupta, T., Aggarwal, M. & Kumar, M. 2017. Comparative Analysis of MAC and HMAC-Sha3 using NS-2. *International Journal of Computer Applications*, 160(6): 9–14.

Gurusinghe, D.R., Kariyawasam, S. & Ouellette, D.S. 2018. Testing of IEC 61850 sampled values based digital substation automation systems. *The Journal of Engineering*, 2018(15): 807–811.

Hamouda, B.E.H.H. 2020. Comparative study of different cryptographic algorithms. *Journal of Information Security*, 11(4): 138–148.

Harbi, Y., Aliouat, Z., Refoufi, A., Harous, S. & Bentaleb, A. 2019. Enhanced authentication and key management scheme for securing data transmission in the internet of things. *Ad Hoc Networks*, 94: 101948. https://doi.org/10.1016/j.adhoc.2019.101948.

Hariri, M. El, Harmon, E., Youssef, T., Saleh, M., Habib, H. & Mohammed, O. 2019. The IEC 61850 sampled measured values protocol: Analysis, threat identification, and feasibility of using NN forecasters to detect spoofed packets †. *Energies*, 12(19).

Harispuru, C. & Schuster, N. Making IEC 61850 GOOSE Communication More Reliable. : 1–6.

Hitachi ABB. FOX615 TEGO1 IEC 61850 GOOSE and Sampled Value Proxy Gateway interface module . FOX615 multiplexing platform . Enabling GOOSE and SV messaging thanks to TEGO1 interface card .

Hodder, S., Kasztenny, B., McGinn, D. & Hunt, R. 2009. IEC 61850 process bus solution addressing business needs of today’s utilities. *2009 Power Systems Conference: Advance Metering, Protection, Control, Communication, and Distributed Resources, PSC 2009*, (March): 56–76.

Hohlbaum, F., Braendle, M. & Alvarez, F. 2010. Cyber security practical considerations for implementing IEC 62351. *PAC World 2010*: 1–8.

Hong, J., Liu, C.C. & Govindarasu, M. 2014. Detection of cyber intrusions using network-based multicast messages for substation automation. *2014 IEEE PES Innovative Smart Grid Technologies Conference, ISGT 2014*, (February).

Hong, S., Shin, D. & Lee, S. Experimenting Security Algorithms for the IEC 61850-based Substation Communication.

Hou, D. & Dolezilek, D. 2010. IEC 61850 – What It Can and Cannot Offer to Traditional Protection Schemes. *SEL Journal of Reliable Power*, 1(2): 1–12.

Hoyos, J., Dehus, M. & Brown, T.X. 2012. Exploiting the GOOSE protocol: A practical attack on cyber-infrastructure. *2012 IEEE Globecom Workshops, GC Wkshps 2012*: 1508–1513.

Hussain, S.M.S., Farooq, S.M. & Ustun, T.S. 2020. A Method for Achieving Confidentiality and Integrity in IEC 61850 GOOSE Messages. *IEEE Transactions on Power Delivery*, 35(5): 2565–2567.

Hussain, S.M.S., Farooq, S.M. & Ustun, T.S. 2019. Analysis and implementation of message authentication code (MAC) algorithms for GOOSE message security. *IEEE Access*, 7: 80980–80984.

Hussain, S.M.S., Ustun, T.S. & Kalam, A. 2020. A Review of IEC 62351 Security Mechanisms for IEC 61850 Message Exchanges. *IEEE Transactions on Industrial Informatics*, 16(9): 5643–5654.

IEC 61850-7-1. 2003. *Communication networks and systems for power utility automation - Part 7-1: Basic communication structure for substation and feeder equipment – Principles and models*.

IEC 61850-7-2. 2003. International Standard International Standard. *61010-1 © Iec:2001*, 2003: 13.

Ingram, D.M.E., Schaub, P., Campbell, D.A. & Taylor, R.R. 2013. Performance analysis of PTP components for IEC 61850 process bus applications. *IEEE Transactions on Instrumentation and Measurement*, 62(4): 710–719.

Ingram, D.M.E., Schaub, P., Taylor, R.R. & Campbell, D.A. 2013. Performance analysis of IEC 61850 sampled value process bus networks. *IEEE Transactions on Industrial Informatics*, 9(3): 1445–1454.

International Electrotechnical Commission. 2009. Communication networks and systems in substations – Part 8-1: Specific Communication Service Mapping (SCSM) – Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3. *International Organisation*, 2007: 1–11.

Ishchenko, D. & Nuqui, R. 2018. Secure Communication of Intelligent Electronic Devices in

Digital Substations. *Proceedings of the IEEE Power Engineering Society Transmission and Distribution Conference*, 2018-April.

Julie, F.G. 2014. *DEVELOPMENT OF AN IEC 61850 STANDARD-BASED AUTOMATION SYSTEM FOR A DISTRIBUTION POWER NETWORK*.

Kalita, L. 2012. Socket programming. *International Journal of Computer Science and Information Technologies*, 5(3): 1812–1822.

Kanabar, M.G. & Sidhu, T.S. 2011. Performance of IEC 61850-9-2 process bus and corrective measure for digital relaying. *IEEE Transactions on Power Delivery*, 26(2): 725–735.

Kang, D.J., Lee, J.J., Kim, B.H. & Hur, D. 2011. Proposal strategies of key management for data encryption in SCADA network of electric power systems. *International Journal of Electrical Power and Energy Systems*, 33(9): 1521–1526. http://dx.doi.org/10.1016/j.ijepes.2009.03.004.

Karnati, R. 2020. *Security of Process Bus in Digital Substation*. University of Michigan-Dearborn. https://deepblue.lib.umich.edu/handle/2027.42/166307.

Kasztenny, B., Whatley, J., Urden, E.A., Burger, J., Finney, D. & Adamiak, M. 2005. Unanswered questions about IEC 61850. *32nd Annual Western Protective Relay Conference*.

Khali, H., Mehdi, R. & Araar, A. 2016. A System-Level Architecture For Hash Message Authentication Code. , (10): 2016.

Khan, R., McLaughlin, K., Laverty, D. & Sezer, S. 2016. IEEE C37.118-2 synchrophasor communication framework: Overview, cyber vulnerabilities analysis and performance evaluation. *ICISSP 2016 - Proceedings of the 2nd International Conference on Information Systems Security and Privacy*, (April 2017): 167–176.

Kim, J., Kim, Y. & Kim, T. 2013. Implementation of secure GOOSE protocol using HSM. *Applied Mechanics and Materials*, 261–262: 236–241.

Kim, J.C. & Kim, T.H. 2014. Implementatio of Secure IEC 61850 Communication. *Proceedings of the CIRED Workshop 2014*, (322): 11–12.

Kriger, C., Behardien, S. & Retonda-Modiya, J.C. 2013. A detailed analysis of the GOOSE message structure in an IEC 61850 standard-based substation automation system. *International Journal of Computers, Communications and Control*, 8(5): 708–721.

Kumar, S. 2019. A Review On Client-Server Based Applications And Research Opportunity. *International Journal of Recent Scientific Research*, (August).

Kumar, S., Das, N. & Islam, S. 2016. Performance evaluation of a process bus architecture in a zone substation based on IEC 61850-9-2. *Asia-Pacific Power and Energy Engineering Conference, APPEEC*, 2016-Janua(November).

Leszczyna, R. 2018. A review of standards with cybersecurity requirements for smart grid.

Computers and Security, 77: 262–276.

Lin, C.Y. & Nadjm-Tehrani, S. 2018. Understanding IEC-60870-5-104 traffic patterns in SCADA networks. *CPSS 2018 - Proceedings of the 4th ACM Workshop on Cyber-Physical System Security, Co-located with ASIA CCS 2018*: 51–60.

Mguzulwa, N.R. 2018. Investigation of Interoperability of Iec 61850 Protection Functions.

Michail, H.E., Kakarountas, A.P., Milidonis, A. & Goutis, C.E. 2004. Efficient implementation of the Keyed-Hash Message Authentication Code (HMAC) using the SHA-1 hash function. *11th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2004*: 567–570.

Mohagheghi, S., Tournier, J.C., Stoupis, J., Guise, L., Coste, T., Andersen, C.A. & Dall, J. 2011. Applications of IEC 61850 in distribution automation. *2011 IEEE/PES Power Systems Conference and Exposition, PSCE 2011*, (March).

Moreira, N., Molina, E., Lázaro, J., Jacob, E. & Astarloa, A. 2016. Cyber-security in substation automation systems. *Renewable and Sustainable Energy Reviews*, 54: 1552–1562. http://dx.doi.org/10.1016/j.rser.2015.10.124.

National Institute of Standards and Technology. 2012. Nist framework and roadmap for smart grid interoperability standards, release 1.0. *Smart Grid Cybersecurity Guidelines and Interoperability Standards (with DVD)*: 19–133.

Ncube, A.M. 2012. *IEC 61850-9-2 BASED SAMPLED VALUES AND IEC 61850-8-1 GOOSE MESSAGES MAPPING ON AN FPGA PLATFORM*.

Omar Hegazi , Eman Hammad , Abdallah Farraj, and D.K. 2017. IEC-61850 GOOSE TRAFFIC MODELING AND GENERATION Omar Hegazi , Eman Hammad , Abdallah Farraj , and Deepa Kundur Department of Electrical and Computer Engineering , University of Toronto , Canada Email : omar.hegazi@mail.utoronto.ca , { ehammad , abdallah ,. , (Xml): 1100–1104.

Oszywa, W. & Gliwa, R. 2012. Combining message encryption and authentication. *Annales UMCS, Informatica*, 11(2): 61–79.

Oyelade, J., Isewon, I., Oladipupo, O. & 2, A.F. 2015. Implementation of Secured Message Transmission using DES and RSA Cryptosystem. *Covenant Journal of Informatics and Communication Technology*, 2(January).

Ozansoy, C.R., Zayegh, A. & Kalam, K. 2007. The real-time publisher/subscriber communication model for distributed substation systems. *IEEE Transactions on Power Delivery*, 22(3): 1411–1423.

Pal, A. & Dash, R. 2015. A Paradigm Shift in Substation Engineering: IEC 61850 Approach. *Procedia Technology*, 21: 8–14. http://dx.doi.org/10.1016/j.protcy.2015.10.003.

Pathan, E.E. & Asad, E.M. 2016. ACSE GOOSE Messages Deployment for IEC61850 Substation Automation System. *International Journal of Advanced Research in*

Electrical, Electronics and Instrumentation Engineering, 5(1): 254–261.

Piantadosi, G., Marrone, S., Sansone, M. & Sansone, C. 2015. A secure, scalable and versatile multi-layer client–server architecture for remote intelligent data processing. *Journal of Reliable Intelligent Environments*, 1(2–4): 173–187.

Retonda-Modiya, J.-C. 2012. *Development of an Embedded System Actuator Node for*. http://hdl.handle.net/20.500.11838/1162.

Robillard, C. 2018. Network and System Management using IEC 62351-7 in IEC 61850 Substations: Design and Implementation. , (December).

Rodriguez, M., Lazaro, J., Bidarte, U., Jimenez, J. & Astarloa, A. 2021. A fixed-latency architecture to secure GOOSE and sampled value messages in substation systems. *IEEE Access*, 9: 51646–51658.

Schlegel, R., Obermeier, S. & Schneider, J. 2017a. A security evaluation of IEC 62351. *Journal of Information Security and Applications*, 34: 197–204.

Schlegel, R., Obermeier, S. & Schneider, J. 2017b. A security evaluation of IEC 62351. *Journal of Information Security and Applications*, 34(June): 197–204.

Shrestha, A., Silveira, M., Yellajosula, J. & Mutha, S.K. 2021. Understanding the Impacts of Time Synchronization and Network Issues on Protection in Digital Secondary Systems. *PAC World Global Conference 2021*: 1–11.

Skendzic, V., Ender, I. & Zweigle, G. 2007. IEC 61850-9-2 Process Bus and Its Impact on Power System Protection and Control Reliability. *9th Annual Western Power Delivery Automation Conference*: 1–7. https://cdn.selinc.com/assets/Literature/Publications/Technical Papers/6275_Process Bus_VS_20070226_Web.pdf?v=20150812-084500.

Skoff, N.M. 2020. Performance Analysis of Sampled Values- Based Protection in IEC 61850 Process Bus Networks. : 1–79.

Sontowski, M. 2016. Workshop Data Security and Privacy. , (June).

Starck, J., Wimmer, D.W. & Majer, K. 2013. SWITCHGEAR OPTIMIZATION USING IEC 61850-9-2. *22 nd International Conference on Electricity Distribution*, (0668): 10–13.

Strobel, M., Wiedermann, N. & Eckert, C. 2016. Novel weaknesses in IEC 62351 protected Smart Grid control systems. *2016 IEEE International Conference on Smart Grid Communications, SmartGridComm 2016*: 266–270.

Suhail Hussain, S.M., Aftab, M.A., Farooq, S.M., Ali, I., Ustun, T.S. & Konstantinou, C. 2023. An Effective Security Scheme for Attacks on Sample Value Messages in IEC 61850 Automated Substations. *IEEE Open Access Journal of Power and Energy*: 1–1. https://ieeexplore.ieee.org/document/10065529/.

Sun, X., Redfern, D.M. & Aggarwal, P.R.K. 2012. *Protection Performance Study for Secondary Systems with IEC61850*.

Tebekaemi, E. 2016. Designing An IEC 61850 Based Power Distribution Substation Simulation / Emulation Testbed for Cyber-Physical Security Studies. *CYBER 2016 : The First International Conference on Cyber-Technologies and Cyber-Systems*, (c): 41–49.

Tesfay, T.T. & Le Boudec, J.Y. 2018. Experimental comparison of multicast authentication for wide area monitoring systems. *IEEE Transactions on Smart Grid*, 9(5): 4394–4404.

Ustun, T.S. 2021. A critical review of iec 61850 testing tools. *Sustainability (Switzerland)*, 13(11).

Ustun, T.S., Farooq, S.M. & Hussain, S.M.S. 2019. A Novel Approach for Mitigation of Replay and Masquerade Attacks in Smartgrids Using IEC 61850 Standard. *IEEE Access*, 7: 156044–156053. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8873588.

Ustun, T.S., Farooq, S.M. & Suhail Hussain, S.M. 2020. Implementing Secure Routable GOOSE and SV Messages Based on IEC 61850-90-5. *IEEE Access*, 8: 26162–26171.

Wang, N., Yao, R., Liu, Y., Wu, Y. & Mou, D. 2019. A key management method for smart substation. *Energy Procedia*, 156: 337–342. https://doi.org/10.1016/j.egypro.2018.11.152.

Weerathunga, Pubudu Eroshan, W.U. 2012. Security Aspects of Smart Grid Communication. *The School of Graduate and Postdoctoral Studies Western University*, 3(September): 1–47.

Wright, J.G. & Wolthusen, S.D. 2016. Limitations of IEC62351-3's public key management. *Proceedings - International Conference on Network Protocols, ICNP*, 2016-Decem(HotPNS): 1–6.

Xue, M. & Zhu, C. 2009. The socket programming and software design for communication based on client/server. *Proceedings of the 2009 Pacific-Asia Conference on Circuits, Communications and System, PACCS 2009*: 775–777.

Yoo, H. & Shon, T. 2016. Challenges and research directions for heterogeneous cyber-physical system based on IEC 61850: Vulnerabilities, security requirements, and security architecture. *Future Generation Computer Systems*, 61: 128–136. http://dx.doi.org/10.1016/j.future.2015.09.026.

Zhao, P. 2012. IEC 61850-9-2 Process Bus Communication Interface for Light Weight Merging Unit Testing Environment. : 85.

J. Shen, S. Chang, J. Shen, Q. Liu, X. Sun, A lightweight multi-layer authentication protocol for wireless body area networks, Future Gen. Comput. Syst. 78 (2018) 956–963.

R. Amin, S.H. Islam, G. Biswas, M.S. Obaidat, A robust mutual authentication protocol for wsn with multiple base-stations, Ad Hoc Netw. 75 (2018) 1–18.

A. Mehmood, M.M. Umar, H. Song, Icmds: secure inter-cluster multiple-key distribution scheme for wireless sensor networks, Ad Hoc Netw. 55 (2017) 97–106.

Aticleworld. 2021. Socket programming in c using TCP/IP - Aticleworld. [online] Available at: <https://aticleworld.com/socket-programming-in-c-using-tcpip/> [Accessed 18 June 2021].

GeeksforGeeks. 2019. *Socket Programming in C/C++ - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/socket-programming-cc/> [Accessed 18 June 2021].

Thakkar, J., 2020. *Types of Encryption: 5 Encryption Algorithms & How to Choose the Right One*. [online] Hashed Out by The SSL Store™. Available at: <https://www.thesslstore.com/blog/types-of-encryption-encryption-algorithms-how-to-choose-the-right-one/> [Accessed 24 June 2021].

Hamouda, B., 2020. Comparative Study of Different Cryptographic Algorithms. Journal of Information Security, 11(03), pp.138-148.

S.A. Fatayer, T., 2020. Secure Communication Using Cryptography and Covert Channel. Computer and Network Security.

Pedamkar, P., 2023 Cryptography vs Encryption | 6 Awesome Differences You Should Learn. [online] EDUCBA. Available at: <https://www.educba.com/cryptography-vs-encryption/> [Accessed 24 June 2021].

ClickSSL, 2022. Symmetric vs Asymmetric Encryption – Know the Difference. [online] ClickSSL Blog - Information about SSL Certificates & Infosec. Available at: https://www.clickssl.net/blog/symmetric-encryption-vs-asymmetric-encryption.

Daboul, M., Wasserbauser, V. & Orsagova, J., 2015. Laboratory testing of the communication based protection relays. In 21st Conference student EEICT. Brně, 2015. Fakulta elektrotechniky a komunikačních technologií.

Apostolov, A., 2020. *Time in IEC 61850 based substation protection and control systems | PAC World*. [online] PAC World. Available at: <https://www.pacw.org/time-in-iec-61850-based-substation-protection-and-control-systems> [Accessed 14 October 2021].

Moon, S. & posts by Silver Moon &rarr;, V. all. 2020. How to Code a Server and Client in C with Sockets on Linux - Code Examples - BinaryTides. *BinaryTides*. https://www.binarytides.com/server-client-example-c-sockets-linux/ 10 April 2022.

Moon, S. & posts by Silver Moon &rarr;, V. all. 2020. Socket programming in C on Linux - The Ultimate Guide for Beginners - BinaryTides. *BinaryTides*. https://www.binarytides.com/socket-programming-c-linux-tutorial/ 10 April 2022.

Saxena, S. 2015. A Guide to Using Raw Sockets - open source for you. *Open Source For You*. https://www.opensourceforu.com/2015/03/a-guide-to-using-raw-sockets/ 10 April 2022.

Gutmann, P. 2014. Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). : 1–7.

Carullo, Moreno. 2020. IEC 61850 Meets IEC 62351: Securing GOOSE Power Grid Weaknesses. Nozomi Networks. https://www.nozominetworks.com/blog/iec-61850-meets-iec-62351/ 18 July 2022.

61850security - overview (no date) GitHub. Available at: https://github.com/61850security (Accessed: December 9, 2022).

Anon. 2021. ▷ What is an IEC 104 - IEC 60870-5-104? - iGrid Smart Guide. iGrid T&D. https://www.igrid-td.com/smartguide/communicationprotocols/iec-60870-5-104/.

Massink, M.R. 2016. IEC 61850: Are Your Substations Secure? *Applied Risk*. https://applied-risk.com/resources/iec-61850-are-your-substations-secure 4 February 2024.

APPENDICES

APPENDIX A

TCP Client Source Code

```
/*This file is part of LearnEveryone*/
/*https://www.youtube.com/watch?v=GY_Gy1ob4nA for more information*/
/* Copyright (C) Ajaze Parvez Khan*/
/*This program is not published*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <sys/socket.h>
#include <sys/types.h>

#include <netinet/in.h>
#include <arpa/inet.h>

#define PORT 5555

void main(){

 int clientSocket;
 struct sockaddr_in serverAddr;
 char buf[1024];

 clientSocket=socket(PF_INET,SOCK_STREAM,0);
 printf("Client socket Created Successfully...\n");

 memset(&serverAddr,'\0',sizeof(serverAddr));
 serverAddr.sin_family=AF_INET;
 serverAddr.sin_port=htons(PORT);
 serverAddr.sin_addr.s_addr=inet_addr("127.0.0.1");

 connect(clientSocket,(struct sockaddr*)&serverAddr,sizeof(serverAddr));
 printf("Connected to Server Successfully...\n");

 recv(clientSocket,buf,1024,0);
 printf("Data Received: %s...\n",buf);
 printf("Closing Connection...\n");
}
```

APPENDIX B

TCP Server Source Code

```
/*This file is part of LearnEveryone*/
/*https://www.youtube.com/watch?v=GY_Gy1ob4nA for more information*/
/* Copyright (C) Ajaze Parvez Khan*/
/*This program is not published*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <sys/socket.h>
#include <sys/types.h>

#include <netinet/in.h>
#include <arpa/inet.h>

#define PORT 5555

void main()
{
 int sockfd;
 struct sockaddr_in serverAddr;

 int newSocket;
 struct sockaddr_in newAddr;

 socklen_t addr_size;
 char buf[1024];

 sockfd=socket(PF_INET,SOCK_STREAM,0);
 printf("Server socket Created Successfully...\n");
 memset(&serverAddr,'\0',sizeof(serverAddr));

 serverAddr.sin_family=AF_INET;
 serverAddr.sin_port=htons(PORT);
 serverAddr.sin_addr.s_addr=inet_addr("127.0.0.1");

 bind(sockfd,(struct sockaddr*)&serverAddr,sizeof(serverAddr));
 printf("Bind to Port Number %d\n",4455);

 listen(sockfd,6);
 printf("Listening...\n");

 newSocket=accept(sockfd,(struct sockaddr*)&newAddr,&addr_size);

 strcpy(buf,"HELLO FROM ME");
 send(newSocket,buf,strlen(buf),0);

 printf("Closing Connection...\n");
}
```

APPENDIX C

Plain GOOSE Source Code

```
/*This file is part of Github*/
/*https://github.com/61850security/R-GoSV*/
/*Source code published by (Hussain et al., 2019) */
/*This program is published*/

#include <arpa/inet.h>
#include <linux/if_packet.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <net/if.h>
#include <netinet/ether.h>
#include <unistd.h>


/* enter your intended MAC address*/
#define DEST_MAC0    0xFC
#define DEST_MAC1    0x61
#define DEST_MAC2    0x98
#define DEST_MAC3    0x8A
#define DEST_MAC4    0x1F
#define DEST_MAC5    0x2F


/* destination MAC address: FC 61 98 8A 46 EE */
/* interface name: enter your internname name */
#define IF_NAME     "eth0"
#define B_SIZE     2048
/* IP Header fields */
char ver_hl =0x45;  // version 4 and header length of ip is 20 (5) bytes (1 byte)
char tos = 0x00; // type of service. IP precedence and Differentiated Service code point (1 byte)
char totlen1 = 0x00; // Total length of the packet (header 20 + data 185=205 ) (2 bytes)
char totlen2 = 0xCD;
char identification1 = 0x6B; // unique identification of each packet (2 bytes)
char identification2 = 0xA1;
char frag_off1 = 0x00; //if the packets are fragmented, then this field will be used. (2 bytes)
char frag_off2 = 0x00;
char ttl = 0x80; // time to live (1 byte)
char protocol = 0x11; // next protocol in the sequence UDP (1 byte)
char hdrchks1 = 0x00; // header check sum (2 bytes)
char hdrchks2 = 0x00;


// source IP address (4 bytes)
char srcaddr0 = 0xEF; //239    127 --> 0x7F
char srcaddr1 = 0xBF; //191    0   --> 0x00
char srcaddr2 = 0x69; //105    0   --> 0x00
char srcaddr3 = 0xBA; //186    1   --> 0x01


// destination IP address (4 bytes)
char dstaddr0 = 0xC0; //192
char dstaddr1 = 0xA8; //168
char dstaddr2 = 0x01; //1
char dstaddr3 = 0x03; //3


/* UDP Header fields */
char srcport1 = 0xDA;
char srcport2 = 0xD1;

char dstport1 = 0x00;
char dstport2 = 0x66;

char lengt1= 0x00;
char lengt2 = 0xB9; /* length of udp header 8 + data 177 = 185 */

char chksum1 = 0xD7;
char chksum2 = 0x3B;

/* Session header */
char len_id = 0x01; /* Length identifier according to RFC1240 OSI connectionless transport services over */
char t_id = 0x40; // transport identifier
char session_id = 0xA1; /* Non-tunnlled GOOSE */
char length_id = 0x18;  /*  22 + length id 1 + common header 1 = 24 */
char common_header= 0x80;
char l_id = 0x16; /* length of common session header spdu length 4 + spdu num 4 + version 2 + timeofcurrent key
4 + timeof next key 2 +
                        security algorithm 2 + keyID 4 = 22 bytes */
char spdu_length1=0x00; /* length of entire SPDU =167 bytes */
char spdu_length2=0x00;
char spdu_length3=0x00;
char spdu_length4=0xA7;
char spdu_num1=0x00;    /* spdu unique identification number */
char spdu_num2=0x00;
char spdu_num3=0x00;
char spdu_num4=0x0D;
char ver1=0x00;
char ver2=0x01;
char TimeofCurrentKey1=0x00;
char TimeofCurrentKey2=0x00;
char TimeofCurrentKey3=0x00;
char TimeofCurrentKey4=0x00;
char TimeofNextKey1=0x00;
char TimeofNextKey2=0x00;
char sa1=0x00; /* Encryption algorithms used*/
char sa2=0x00; /* authentication algorithm used */
char keyID1=0x00;
char keyID2=0x00;
char keyID3=0x00;
char keyID4=0x00;
char len1=0x00; /* 4 length + 1 payload type + 1 simulation + 2 APPDI + 2 length + 137 = 147 */
char len2=0x00;
char len3=0x00;
char len4=0x93;
char pl_type=0x81; /* 81 for GOOSE 82 for SV */
char simulation=0x00; /* boolean value*/
char APPID1=0x00;
char APPID2=0x01;
char length1=0x00; /* 137 GOOSE + 2 length = 139*/
char length2=0x8B;


/* GOOSE message according to IEC 61850-8-1*/
char goosePDU_tag1=0x61;                        /* goosePDU tag  */
char goosePDU_tag2=0x81;
char goosePDU_length=0x86;                    /* goosePDU length  */
char gocbRef_tag=0x80;                        /* gocbRef (GOOSE Control Block Reference) tag  */
char gocbRef_length=0x1A;                     /* gocbRef length  */
char gocbRef_value1=0x46;                     /* gocbRef value  */
char gocbRef_value2=0x52;
char gocbRef_value3=0x45;
char gocbRef_value4=0x41;
char gocbRef_value5=0x2D;
char gocbRef_value6=0x47;
char gocbRef_value7=0x6F;
char gocbRef_value8=0x53;
char gocbRef_value9=0x56;
char gocbRef_value10=0x2D;
char gocbRef_value11=0x31;
char gocbRef_value12=0x20;
char gocbRef_value13=0x2F;
char gocbRef_value14=0x4C;
char gocbRef_value15=0x4C;
char gocbRef_value16=0x4E;
char gocbRef_value17=0x30;
char gocbRef_value18=0x24;
char gocbRef_value19=0x47;
char gocbRef_value20=0x4F;
char gocbRef_value21=0x24;
char gocbRef_value22=0x67;
char gocbRef_value23=0x63;
char gocbRef_value24=0x62;
char gocbRef_value25=0x30;
char gocbRef_value26=0x31;
char timeAllowedtoLive_tag=0x81;   /* timeAllowedtoLive tag  */
char timeAllowedtoLive_length=0x03;           /* timeAllowedtoLive length  */
char timeAllowedtoLive_value1=0x00;           /* timeAllowedtoLive value  */
char timeAllowedtoLive_value2=0x9C;
char timeAllowedtoLive_value3=0x40;
char dataset_tag=0x82;                        /* data set tag*/
char dataset_length=0x18;             /* data set length*/
char dataset_value1=0x46;                     /* data set value*/
char dataset_value2=0x52;
char dataset_value3=0x45;
char dataset_value4=0x41;
char dataset_value5=0x2D;
char dataset_value6=0x47;
char dataset_value7=0x6F;
char dataset_value8=0x53;
char dataset_value9=0x56;
char dataset_value10=0x2D;
char dataset_value11=0x31;
char dataset_value12=0x20;
char dataset_value13=0x2F;
char dataset_value14=0x4C;
char dataset_value15=0x4C;
char dataset_value16=0x4E;
char dataset_value17=0x30;
char dataset_value18=0x24;
char dataset_value19=0x47;
char dataset_value20=0x4F;
char dataset_value21=0x4F;
char dataset_value22=0x53;
char dataset_value23=0x45;
char dataset_value24=0x31;
char goID_tag=0x83;                           /* goID tag*/
char goID_length=0x0B;                        /* goID length*/
char goID_value1=0x46;                    /* goID value [11]*/
char goID_value2=0x52;
char goID_value3=0x45;
char goID_value4=0x41;
char goID_value5=0x2D;
char goID_value6=0x47;
char goID_value7=0x6F;
char goID_value8=0x53;
char goID_value9=0x56;
char goID_value10=0x2D;
char goID_value11=0x31;
char time_tag=0x84;                       /* time tag*/
char time_length=0x08;                    /* time length*/
char time_value1=0x38;                    /* time value*/
char time_value2=0x6E;
char time_value3=0xBB;
char time_value4=0xF3;
char time_value5=0x42;
char time_value6=0x17;
char time_value7=0x28;
char time_value8=0x0A;                    /* st_Num (State Number) tag */
char st_Num_tag=0x85;                     /* st_Num length */
char st_Num_length=0x01;          /* st_Num value */
char st_Num_value=0x01;
char sq_Num_tag=0x86;                     /* sq_Num (sequence Number) tag */
char sq_Num_length=0x01;                  /* sq_Num length */
char sq_Num_value=0x0A;                           /* sq_Num value */
char test_tag=0x87;                       /*test tag*/
char test_length=0x01;                    /*test length*/
char test_value=0x00;                     /*test value*/
char confRev_tag=0x88;                    /*confRev (Configuration Revision) tag*/
char confRev_length=0x01;                 /*confRev length*/
char confRev_value=0x01;          /*confRev value*/
char ndsCom_tag=0x89;                     /*ndsCom (needs Commissioning) tag*/
char ndsCom_length=0x01;                  /*ndsCom length*/
char ndsCom_value=0x00;                   /*ndsCom value*/
char numDatSetEntries_tag=0x8A;           /* number of members of Data Set */
char numDatSetEntries_length=0x01;
char numDatSetEntries_value=0x08;
char alldata_tag=0xAB;                    /*all data*/
char alldata_length=0x20;
char alldata_value1=0x83;
char alldata_value2=0x01;
char alldata_value3=0x00;
char alldata_value4=0x84;
char alldata_value6=0x03;
char alldata_value5=0x03;
char alldata_value7=0x00;
char alldata_value8=0x00;
char alldata_value9=0x83;
char alldata_value10=0x01;
char alldata_value11=0x00;
char alldata_value12=0x84;
char alldata_value13=0x03;
char alldata_value14=0x03;
char alldata_value15=0x00;
char alldata_value16=0x00;
char alldata_value17=0x83;
char alldata_value18=0x01;
char alldata_value19=0x00;
char alldata_value20=0x84;
char alldata_value21=0x03;
char alldata_value22=0x03;
char alldata_value23=0x00;
char alldata_value24=0x00;
char alldata_value25=0x83;
char alldata_value26=0x01;
char alldata_value27=0x00;
char alldata_value28=0x84;
char alldata_value29=0x03;
char alldata_value30=0x03;
char alldata_value31=0x00;
char alldata_value32=0x00;


unsigned char goosedata[137] =
{
0x61, 0x81, 0x86, 0x80, 0x1A, 0x46, 0x52, 0x45, 0x41, 0x2D, 0x47, 0x6F, 0x53, 0x56, 0x2D, 0x31, 0x20, 0x2F,
0x4C, 0x4C, 0x4E, 0x30, 0x24, 0x47, 0x4F, 0x24, 0x67, 0x63, 0x62, 0x30, 0x31, 0x81, 0x03, 0x00, 0x9C, 0x40,
0x82, 0x18, 0x46, 0x52, 0x45, 0x41, 0x2D, 0x47, 0x6F, 0x53, 0x56, 0x2D, 0x31, 0x20, 0x2F, 0x4C, 0x4C, 0x4E,
0x30, 0x24, 0x47, 0x4F, 0x4F, 0x53, 0x45, 0x31, 0x83, 0x0B, 0x46, 0x52, 0x45, 0x41, 0x2D, 0x47, 0x6F, 0x53,
0x56, 0x2D, 0x31, 0x84, 0x08, 0x38, 0x6E, 0xBB, 0xF3, 0x42, 0x17, 0x28, 0x0A, 0x85, 0x01, 0x01, 0x86, 0x01,
0x0A, 0x87, 0x01, 0x00, 0x88, 0x01, 0x01, 0x89, 0x01, 0x00, 0x8A, 0x01, 0x08, 0xAB, 0x20, 0x83, 0x01, 0x00,
0x84, 0x03, 0x03, 0x00, 0x00, 0x83, 0x01, 0x00, 0x84, 0x03, 0x03, 0x00, 0x00, 0x83, 0x01, 0x00, 0x84, 0x03,
0x03, 0x00, 0x00, 0x83, 0x01, 0x00, 0x84, 0x03, 0x03, 0x00, 0x00
};


char signature=0x85;
char sig_len=0x00; /* length of the signature value 32 bytes generated by HMAC-SHA256*/


int main(int argc, char *argv[])
{
   int sfd;
   int i=0,j=0;
   struct ifreq if_idx;
   struct ifreq if_mac;
   int tx_len;
   unsigned char sendbuf[B_SIZE];
   struct sockaddr_ll socket_address; /* The sockaddr_ll structure is a device-independent physical-layer
address.*/
   char ifName[IFNAMSIZ];


   /* Get interface name */
   strcpy(ifName, IF_NAME);

   /* Open RAW socket to send on */
   if ((sfd = socket(AF_PACKET, SOCK_RAW, IPPROTO_RAW)) == -1)
   {
      perror("socket");
   }

   /* clear the struct ifreq if_idx with memset system call */
   memset(&if_idx, 0, sizeof(struct ifreq));

   /* copy interface name into struct ifreq if_idx */
   strncpy(if_idx.ifr_name, ifName, IFNAMSIZ-1);

   /* configure the interface index */
   if (ioctl(sfd, SIOCGIFINDEX, &if_idx) < 0)
      perror("SIOCGIFINDEX");

   // Loop forever
```

```
while(1) {


    /* Buffer of BUF_SIZ bytes we'll construct our frame in.
       First, clear it all to zero. */
    memset(sendbuf, 0, B_SIZE);
    tx_len = 0;

    /* Construct the UDP header */

    /* Destination MAC address */
    sendbuf[tx_len++] = DEST_MAC0;
    sendbuf[tx_len++] = DEST_MAC1;
    sendbuf[tx_len++] = DEST_MAC2;
    sendbuf[tx_len++] = DEST_MAC3;
    sendbuf[tx_len++] = DEST_MAC4;
    sendbuf[tx_len++] = DEST_MAC5;

          //source address MAC FC 61 98 EA BC 20

       /* Source MAC address */
    sendbuf[tx_len++] = 0xA0;
    sendbuf[tx_len++] = 0xB3;
    sendbuf[tx_len++] = 0xCC;
    sendbuf[tx_len++] = 0xC5;
    sendbuf[tx_len++] = 0x77;
    sendbuf[tx_len++] = 0xA1;


    /* Ethertype field IP protocol */
    sendbuf[tx_len++] = 0x08;
    sendbuf[tx_len++] = 0x00;

    /*  PDU fields */
    sendbuf[tx_len++] = ver_hl;
    sendbuf[tx_len++] = tos;
    sendbuf[tx_len++] = totlen1;
    sendbuf[tx_len++] = totlen2;
    sendbuf[tx_len++] = identification1;
          sendbuf[tx_len++] = identification2;
          sendbuf[tx_len++] = frag_off1;
    sendbuf[tx_len++] = frag_off2;
    sendbuf[tx_len++] = ttl;
    sendbuf[tx_len++] = protocol;
    sendbuf[tx_len++] = hdrchks1;
    sendbuf[tx_len++] = hdrchks2;
    sendbuf[tx_len++] = srcaddr0;
    sendbuf[tx_len++] = srcaddr1;
    sendbuf[tx_len++] = srcaddr2;
    sendbuf[tx_len++] = srcaddr3;
    sendbuf[tx_len++] = dstaddr0;
    sendbuf[tx_len++] = dstaddr1;
    sendbuf[tx_len++] = dstaddr2;
    sendbuf[tx_len++] = dstaddr3;
    sendbuf[tx_len++] = srcport1;
    sendbuf[tx_len++] = srcport2;
    sendbuf[tx_len++] = dstport1;
          sendbuf[tx_len++] = dstport2;
          sendbuf[tx_len++] = lengt1;
          sendbuf[tx_len++] = lengt2;
          sendbuf[tx_len++] = chksum1;
          sendbuf[tx_len++] = chksum2;
```

```
sendbuf[tx_len++] = len_id;
          sendbuf[tx_len++] = t_id;
sendbuf[tx_len++] = session_id;
          sendbuf[tx_len++] = length_id;
          sendbuf[tx_len++] = common_header;
          sendbuf[tx_len++] = l_id;
          sendbuf[tx_len++] = spdu_length1;
          sendbuf[tx_len++] = spdu_length2;
          sendbuf[tx_len++] = spdu_length3;
          sendbuf[tx_len++] = spdu_length4;
sendbuf[tx_len++] = spdu_num1;
          sendbuf[tx_len++] = spdu_num2;
          sendbuf[tx_len++] = spdu_num3;
          sendbuf[tx_len++] = spdu_num4;
          sendbuf[tx_len++] = ver1;
          sendbuf[tx_len++] = ver2;
sendbuf[tx_len++] = TimeofCurrentKey1;
sendbuf[tx_len++] = TimeofCurrentKey2;
sendbuf[tx_len++] = TimeofCurrentKey3;
          sendbuf[tx_len++] = TimeofCurrentKey4;
sendbuf[tx_len++] = TimeofNextKey1;
sendbuf[tx_len++] = TimeofNextKey2;
          sendbuf[tx_len++] = sa1;
          sendbuf[tx_len++] = sa2;
          sendbuf[tx_len++] = keyID1;
          sendbuf[tx_len++] = keyID2;
          sendbuf[tx_len++] = keyID3;
          sendbuf[tx_len++] = keyID4;
sendbuf[tx_len++] = len1;
          sendbuf[tx_len++] = len2;
          sendbuf[tx_len++] = len3;
          sendbuf[tx_len++] = len4;
          sendbuf[tx_len++] = pl_type;
          sendbuf[tx_len++] = simulation; /* boolean value*/
          sendbuf[tx_len++] = APPID1;
          sendbuf[tx_len++] = APPID2;
          sendbuf[tx_len++] = length1;
          sendbuf[tx_len++] = length2;


          for(j=0;j<137;j++)
              sendbuf[tx_len++] = goosedata[j];

          sendbuf[tx_len++] = signature;
          sendbuf[tx_len++] = sig_len;

          sendbuf[tx_len++] = 0xB7;
          sendbuf[tx_len++] = 0x09;
          sendbuf[tx_len++] = 0xD7;
          sendbuf[tx_len++] = 0x83;

          /* Index of the network device */
    socket_address.sll_ifindex = if_idx.ifr_ifindex;  /* Network Interface number */

    /* Address length*/
    socket_address.sll_halen = ETH_ALEN; /* Length of Ethernet address */

    /* Destination MAC */
    socket_address.sll_addr[0] = DEST_MAC0;
    socket_address.sll_addr[1] = DEST_MAC1;
    socket_address.sll_addr[2] = DEST_MAC2;
```

```
        socket_address.sll_addr[3] = DEST_MAC3;
        socket_address.sll_addr[4] = DEST_MAC4;
        socket_address.sll_addr[5] = DEST_MAC5;

        /* Send packet */
        if (sendto(sfd, sendbuf, tx_len, 0, (struct sockaddr*)&socket_address, sizeof(struct sockaddr_ll)) < 0)
            printf("Send failed\n");
        else {
            printf("Sent :");
            for (i=0; i < tx_len; i++)
                printf("%02x:", sendbuf[i]);
            printf("\n");
        }
        /* Wait specified number of microseconds
           1,000,000 microseconds = 1 second
           */
        usleep(1000000);
    }
    return 0;
}
```

APPENDIX D

R-GOOSE Source Code

```
/*This file is part of Github*/
/*https://github.com/61850security/R-GoSV*/
/* Source code published by (Hussain et al., 2019) */
/*This program is published*/


#include <arpa/inet.h>
#include <linux/if_packet.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <net/if.h>
#include <netinet/ether.h>
#include <unistd.h>
#include <openssl/hmac.h>
#include <openssl/evp.h>

/* Keep your intended destination MAC here*/
#define DEST_MAC0    0xFC
#define DEST_MAC1    0x61
#define DEST_MAC2    0x98
#define DEST_MAC3    0x8A
#define DEST_MAC4    0x1F
#define DEST_MAC5    0x2F

#define IF_NAME     "eth0"
#define B_SIZE     2048
/* IP Header fields */
char ver_hl =0x45;  // version 4 and header length of ip is 20 (5) bytes (1 byte)
char tos = 0x00; // type of service. IP precedence and Differentiated Service code point (1 byte)
char totlen1 = 0x00; // Total length of the packet (header 20 + udp packet size 217) (2 bytes)
char totlen2 = 0xED;
char identification1 = 0x6B; // unique identification of each packet (2 bytes)
char identification2 = 0xA1;
char frag_off1 = 0x00; //if the packets are fragmented, then this field will be used. (2 bytes)
char frag_off2 = 0x00;
char ttl = 0x80; // time to live (1 byte)
char protocol = 0x11; // next protocol in the sequence UDP (1 byte)
char hdrchks1 = 0x00; // header check sum (2 bytes)
char hdrchks2 = 0x00;
```

```
// source IP address (4 bytes)
char srcaddr0 = 0xEF; //239    127 --> 0x7F
char srcaddr1 = 0xBF; //191    0   --> 0x00
char srcaddr2 = 0x69; //105    0   --> 0x00
char srcaddr3 = 0xBA; //186    1   --> 0x01

// destination IP address (4 bytes)
char dstaddr0 = 0xC0; //192
char dstaddr1 = 0xA8; //168
char dstaddr2 = 0x01; //1
char dstaddr3 = 0x03; //3

/* UDP Header fields */
char srcport1 = 0xDA;
char srcport2 = 0xD1;

char dstport1 = 0x00;
char dstport2 = 0x66;

char lengt1= 0x00;
char lengt2 = 0xD9; /* length of udp header 8 + udp pay load size = 209 (38+137+34) = 217 */

char chksum1 = 0xD7;
char chksum2 = 0x3B;
/* Session header */
char len_id = 0x01; /* Length identifier according to RFC1240 OSI connectionless transport services over */
char t_id = 0x40; // transport identifier
char session_id = 0xA1; /* Non-tunnlled goose */
char length_id = 0x18;  /*  length of common session header 22  + length id 1 + common header 1 = 24 */
char common_header= 0x80;
char l_id = 0x16; /* length of common session header spdu length 4 + spdu num 4 + version 2 + timeofcurrent key
4 + timeofnextkey 2 +
                        security algorithm 2 + keyID 4 = 22 bytes */
char spdu_length1=0x00; /* length of entire SPDU =199 bytes */
char spdu_length2=0x00; /* spdu num 4 + ver 2 + TofCkey 4 + TofNKey 2 + SA 2 + keyID 4 + len 4 + pl_type 1 +
simulation 1  */
char spdu_length3=0x00; /* APPID 2 + length 2 + SV size  + signature 1 + sig_len 1 + sign_val 32 =199 */
char spdu_length4=0xC7;
char spdu_num1=0x00;    /* spdu unique identification number */
char spdu_num2=0x00;
char spdu_num3=0x00;
char spdu_num4=0x0D;
char ver1=0x00;
char ver2=0x01;
char TimeofCurrentKey1=0x5B; /*hexadecimal timestamp/epoch */
char TimeofCurrentKey2=0xFC;  /* Tuesday, November 27, 2018 4:48:00 PM */
char TimeofCurrentKey3=0xF6;
char TimeofCurrentKey4=0xB0;
char TimeofNextKey1=0x00;
char TimeofNextKey2=0x3C; /* 60 minutes for time of next key */
char sa1=0x00; /* Encryption algorithms used*/
char sa2=0x03; /* authentication algorithm used */
char keyID1=0x00;
char keyID2=0x00;
char keyID3=0x00;
char keyID4=0x0C;
char len1=0x00; /* 1 payload type + 1 simulation + 2 APPDI + 2 length + goose data size 137 =143 */
char len2=0x00;
char len3=0x00;
char len4=0x8F;
char pl_type=0x81; /* 81 for GOOSE 82 for SV */
char simulation=0x01; /* boolean value*/
char APPID1=0x00;
char APPID2=0x01;
char length1=0x00; /* goose data size 137 */
char length2=0x89;
/* GOOSE message according to IEC 61850-8-1*/
char goosePDU_tag1=0x61;                /* goosePDU tag  */
char goosePDU_tag2=0x81;
char goosePDU_length=0x86;              /* goosePDU length  */
char gocbRef_tag=0x80;                  /* gocbRef (GOOSE Control Block Reference) tag  */
char gocbRef_length=0x1A;               /* gocbRef length  */
char gocbRef_value1=0x46;               /* gocbRef value  */
char gocbRef_value2=0x52;
char gocbRef_value3=0x45;
char gocbRef_value4=0x41;
char gocbRef_value5=0x2D;
char gocbRef_value6=0x47;
char gocbRef_value7=0x6F;
char gocbRef_value8=0x53;
char gocbRef_value9=0x56;
char gocbRef_value10=0x2D;
char gocbRef_value11=0x31;
char gocbRef_value12=0x20;
char gocbRef_value13=0x2F;
char gocbRef_value14=0x4C;
char gocbRef_value15=0x4C;
char gocbRef_value16=0x4E;
char gocbRef_value17=0x30;
char gocbRef_value18=0x24;
char gocbRef_value19=0x47;
char gocbRef_value20=0x4F;
char gocbRef_value21=0x24;
char gocbRef_value22=0x67;
char gocbRef_value23=0x63;
char gocbRef_value24=0x62;
char gocbRef_value25=0x30;
char gocbRef_value26=0x31;
char timeAllowedtoLive_tag=0x81;   /* timeAllowedtoLive tag  */
char timeAllowedtoLive_length=0x03;          /* timeAllowedtoLive length  */
char timeAllowedtoLive_value1=0x00;          /* timeAllowedtoLive value  */
char timeAllowedtoLive_value2=0x9C;
char timeAllowedtoLive_value3=0x40;
char dataset_tag=0x82;                       /* data set tag*/
char dataset_length=0x18;          /* data set length*/
char dataset_value1=0x46;                    /* data set value*/
char dataset_value2=0x52;
char dataset_value3=0x45;
char dataset_value4=0x41;
char dataset_value5=0x2D;
char dataset_value6=0x47;
char dataset_value7=0x6F;
char dataset_value8=0x53;
char dataset_value9=0x56;
char dataset_value10=0x2D;
char dataset_value11=0x31;
char dataset_value12=0x20;
char dataset_value13=0x2F;
char dataset_value14=0x4C;
char dataset_value15=0x4C;
char dataset_value16=0x4E;
char dataset_value17=0x30;
char dataset_value18=0x24;
char dataset_value19=0x47;
char dataset_value20=0x4F;
char dataset_value21=0x4F;
char dataset_value22=0x53;
char dataset_value23=0x45;
char dataset_value24=0x31;
char goID_tag=0x83;                          /* goID tag*/
char goID_length=0x0B;                       /* goID length*/
char goID_value1=0x46;                       /* goID value [11]*/
char goID_value2=0x52;
char goID_value3=0x45;
char goID_value4=0x41;
char goID_value5=0x2D;
char goID_value6=0x47;
char goID_value7=0x6F;
char goID_value8=0x53;
char goID_value9=0x56;
char goID_value10=0x2D;
char goID_value11=0x31;
char time_tag=0x84;                       /* time tag*/
char time_length=0x08;                    /* time length*/
char time_value1=0x38;                    /* time value*/
char time_value2=0x6E;
char time_value3=0xBB;
char time_value4=0xF3;
char time_value5=0x42;
char time_value6=0x17;
char time_value7=0x28;
char time_value8=0x0A;                    /* st_Num (State Number) tag */
char st_Num_tag=0x85;                     /* st_Num length */
char st_Num_length=0x01;                  /* st_Num value */
char st_Num_value=0x01;
char sq_Num_tag=0x86;                     /* sq_Num (sequence Number) tag */
char sq_Num_length=0x01;                  /* sq_Num length */
char sq_Num_value=0x0A;                           /* sq_Num value */
char test_tag=0x87;                       /*test tag*/
char test_length=0x01;                    /*test length*/
char test_value=0x00;                     /*test value*/
char confRev_tag=0x88;                    /*confRev (Configuration Revision) tag*/
char confRev_length=0x01;                 /*confRev length*/
char confRev_value=0x01;                  /*confRev value*/
char ndsCom_tag=0x89;                     /*ndsCom (needs Commissioning) tag*/
char ndsCom_length=0x01;                  /*ndsCom length*/
char ndsCom_value=0x00;                   /*ndsCom value*/
char numDatSetEntries_tag=0x8A;           /* number of members of Data Set */
char numDatSetEntries_length=0x01;
char numDatSetEntries_value=0x08;
char alldata_tag=0xAB;                    /*all data*/
char alldata_length=0x20;
char alldata_value1=0x83;
char alldata_value2=0x01;
char alldata_value3=0x00;
char alldata_value4=0x84;
char alldata_value6=0x03;
char alldata_value5=0x03;
char alldata_value7=0x00;
char alldata_value8=0x00;
char alldata_value9=0x83;
char alldata_value10=0x01;
char alldata_value11=0x00;
char alldata_value12=0x84;
char alldata_value13=0x03;
char alldata_value14=0x03;
char alldata_value15=0x00;
char alldata_value16=0x00;
char alldata_value17=0x83;
char alldata_value18=0x01;
char alldata_value19=0x00;
char alldata_value20=0x84;
char alldata_value21=0x03;
char alldata_value22=0x03;
char alldata_value23=0x00;
char alldata_value24=0x00;
char alldata_value25=0x83;
char alldata_value26=0x01;
char alldata_value27=0x00;
char alldata_value28=0x84;
char alldata_value29=0x03;
char alldata_value30=0x03;
char alldata_value31=0x02;
char alldata_value32=0x01;

unsigned char goosedata[137] =
 {
0x61, 0x81, 0x86, 0x80, 0x1A, 0x46, 0x52, 0x45, 0x41, 0x2D, 0x47, 0x6F, 0x53, 0x56, 0x2D,
0x31, 0x20, 0x2F, 0x4C, 0x4C, 0x4E, 0x30, 0x24, 0x47, 0x4F, 0x24, 0x67, 0x63, 0x62, 0x30,
0x31, 0x81, 0x03, 0x00, 0x9C, 0x40, 0x82, 0x18, 0x46, 0x52, 0x45, 0x41, 0x2D, 0x47, 0x6F,
0x53, 0x56, 0x2D, 0x31, 0x20, 0x2F, 0x4C, 0x4C, 0x4E, 0x30, 0x24, 0x47, 0x4F, 0x4F, 0x53,
0x45, 0x31, 0x83, 0x0B, 0x46, 0x52, 0x45, 0x41, 0x2D, 0x47, 0x6F, 0x53, 0x56, 0x2D, 0x31,
0x84, 0x08, 0x38, 0x6E, 0xBB, 0xF3, 0x42, 0x17, 0x28, 0x0A, 0x85, 0x01, 0x01, 0x86, 0x01,
0x0A, 0x87, 0x01, 0x00, 0x88, 0x01, 0x01, 0x89, 0x01, 0x00, 0x8A, 0x01, 0x08, 0xAB, 0x20,
0x83, 0x01, 0x00, 0x84, 0x03, 0x03, 0x00, 0x00, 0x83, 0x01, 0x00, 0x84, 0x03, 0x03, 0x00,
0x00, 0x83, 0x01, 0x00, 0x84, 0x03, 0x03, 0x00, 0x00, 0x83, 0x01, 0x00, 0x84, 0x03, 0x03,
0x02, 0x01
};
unsigned char signature_data[173]=
{
0xA1, 0x18, 0x80, 0x16, 0x00, 0x00, 0x00, 0xC7, 0x00, 0x00, 0x00, 0x0D, 0x00, 0x01, 0x5B, 0xFC,
0xF6, 0xB0, 0x00, 0x3C, 0x02, 0x03, 0x00, 0x00, 0x00, 0x0C, 0x00, 0x00, 0x00, 0x8F, 0x81, 0x01,
0x00, 0x01, 0x00, 0x89, 0x61, 0x81, 0x86, 0x80, 0x1A, 0x46, 0x52, 0x45, 0x41, 0x2D, 0x47, 0x6F,
0x53, 0x56, 0x2D, 0x31, 0x20, 0x2F, 0x4C, 0x4C, 0x4E, 0x30, 0x24, 0x47, 0x4F, 0x24, 0x67, 0x63,
0x62, 0x30, 0x31, 0x81, 0x03, 0x00, 0x9C, 0x40, 0x82, 0x18, 0x46, 0x52, 0x45, 0x41, 0x2D, 0x47,
0x6F, 0x53, 0x56, 0x2D, 0x31, 0x20, 0x2F, 0x4C, 0x4C, 0x4E, 0x30, 0x24, 0x47, 0x4F, 0x4F, 0x53,
0x45, 0x31, 0x83, 0x0B, 0x46, 0x52, 0x45, 0x41, 0x2D, 0x47, 0x6F, 0x53, 0x56, 0x2D, 0x31, 0x84,
0x08, 0x38, 0x6E, 0xBB, 0xF3, 0x42, 0x17, 0x28, 0x0A, 0x85, 0x01, 0x01, 0x86, 0x01, 0x0A, 0x87,
0x01, 0x00, 0x88, 0x01, 0x01, 0x89, 0x01, 0x00, 0x8A, 0x01, 0x08, 0xAB, 0x20, 0x83, 0x01, 0x00,
0x84, 0x03, 0x03, 0x00, 0x00, 0x83, 0x01, 0x00, 0x84, 0x03, 0x03, 0x00, 0x00, 0x83, 0x01, 0x00,
0x84, 0x03, 0x03, 0x00, 0x00, 0x83, 0x01, 0x00, 0x84, 0x03, 0x03, 0x02, 0x01
};  //updated
unsigned char ciphertext[173]=
{
0xA3, 0x50, 0x43, 0x00, 0xf3, 0x8f, 0x23, 0xb5, 0x3e, 0x07, 0x34, 0x99, 0x75, 0xd8, 0xbe, 0x6f,
0x7c, 0x66, 0x48, 0x6c, 0xa9, 0xe0, 0x7e, 0x8e, 0x7f, 0x3f, 0x16, 0x02, 0x31, 0x7f, 0x28, 0x4e,
0xdc, 0x48, 0xff, 0x26, 0x99, 0xd5, 0x28, 0xa7, 0x22, 0x54, 0x0e, 0x0a, 0x19, 0x93, 0xa4, 0xb7,
0x56, 0x52, 0xAD, 0xC9, 0x68, 0x3C, 0x82, 0xFD, 0xAF, 0x30, 0xAF, 0x7D, 0x48, 0xE8, 0x8B, 0x8E,
0x9B, 0x7C, 0xF9, 0x1C, 0x27, 0x0B, 0x05, 0xB8, 0x2B, 0xF7, 0x8D, 0xF2, 0x4F, 0x0D, 0xAB, 0x4D,
0x57, 0xDA, 0x29, 0x0B, 0x7E, 0x75, 0x24, 0x24, 0x9A, 0x7C, 0xAA, 0x7A, 0x38, 0x7C, 0x1C, 0xAA,
0x8F, 0x39, 0xA8, 0xE3, 0x74, 0xE5, 0xBB, 0x2F, 0x6A, 0x9E, 0x39, 0xB0, 0x32, 0x9D, 0x56, 0x7E,
0xA8, 0x83, 0xE3, 0xF8, 0xEB, 0x2E, 0x97, 0xD8, 0x04, 0xA0, 0x4E, 0x56, 0x5D, 0xC6, 0xC2, 0xBB,
0x5B, 0x26, 0x7C, 0x5E, 0x91, 0x41, 0x97, 0x08, 0x15, 0x30, 0x6D, 0x8E, 0x5F, 0x0E, 0x90, 0x13,
0xC4, 0x8F, 0xF9, 0x60, 0x66, 0x84, 0x08, 0x30, 0xD9, 0xFB, 0xBA, 0x97, 0x44, 0xC1, 0xBE, 0xA3,
0xAD, 0xA6, 0x37, 0x75, 0x18, 0xF6, 0xDB, 0xD8, 0x3E, 0x4B, 0xAA, 0x82, 0x21
};
char signature=0x85;
char sig_len=0x20; /* length of the signature value 32 bytes generated by HMAC-SHA256*/
char sig_val[32]= /* It is calculated from session identifier(SI) to end of user data payload */
{
0x94, 0x86, 0x5b, 0x96, 0xbf, 0xc8, 0x8d, 0x25, 0x29, 0xd1, 0x15, 0x24, 0xd2, 0x2b, 0xcb, 0x62,
0x58, 0xa5, 0x61, 0x9a, 0x81, 0x91, 0x99, 0x25, 0x06, 0x88, 0x0b, 0x96, 0xfa, 0x9c, 0x6a, 0x8c
 };
int main(int argc, char *argv[])
{
   int sfd;
   int i=0,j=0;
   struct ifreq if_idx;
   struct ifreq if_mac;
   int tx_len;
   unsigned char sendbuf[B_SIZE],Data[173];
   struct sockaddr_ll socket_address; /* The sockaddr_ll structure is a device-independent physical-layer
address.*/
   char ifName[IFNAMSIZ];
   unsigned char key[14]= { 0x32, 0x21, 0x23, 0x52, 0x71, 0x98, 0x24, 0x03, 0x38, 0x27, 0x01, 0x12, 0x95, 0x23};
   unsigned char *hash;

   /* Get interface name */
   strcpy(ifName, IF_NAME);
```

```
/* Open RAW socket to send on */
if ((sfd = socket(AF_PACKET, SOCK_RAW, IPPROTO_RAW)) == -1)
{
    perror("socket");
}

/* clear the struct ifreq if_idx with memset system call */
memset(&if_idx, 0, sizeof(struct ifreq));

/* copy interface name into struct ifreq if_idx */
strncpy(if_idx.ifr_name, ifName, IFNAMSIZ-1);

/* configure the interface index */
if (ioctl(sfd, SIOCGIFINDEX, &if_idx) < 0)
    perror("SIOCGIFINDEX");

// Loop forever
while(1) {
    /* Buffer of BUF_SIZ bytes we'll construct our frame in.
       First, clear it all to zero. */
    memset(sendbuf, 0, B_SIZE);
    tx_len = 0;

    /* Construct the UDP header */

    /* Destination MAC address */
    sendbuf[tx_len++] = DEST_MAC0;
    sendbuf[tx_len++] = DEST_MAC1;
    sendbuf[tx_len++] = DEST_MAC2;
    sendbuf[tx_len++] = DEST_MAC3;
    sendbuf[tx_len++] = DEST_MAC4;
    sendbuf[tx_len++] = DEST_MAC5;

    /* Source MAC address */
    sendbuf[tx_len++] = 0xA0;
    sendbuf[tx_len++] = 0xB3;
    sendbuf[tx_len++] = 0xCC;
    sendbuf[tx_len++] = 0xC5;
    sendbuf[tx_len++] = 0x77;
    sendbuf[tx_len++] = 0xA1;

    /* Ethertype field IP protocol */
    sendbuf[tx_len++] = 0x08;
    sendbuf[tx_len++] = 0x00;
```

```
/*  PDU fields */
sendbuf[tx_len++] = ver_hl;
sendbuf[tx_len++] = tos;
sendbuf[tx_len++] = totlen1;
sendbuf[tx_len++] = totlen2;
sendbuf[tx_len++] = identification1;
      sendbuf[tx_len++] = identification2;
      sendbuf[tx_len++] = frag_off1;
sendbuf[tx_len++] = frag_off2;
sendbuf[tx_len++] = ttl;
sendbuf[tx_len++] = protocol;
sendbuf[tx_len++] = hdrchks1;
sendbuf[tx_len++] = hdrchks2;
sendbuf[tx_len++] = srcaddr0;
sendbuf[tx_len++] = srcaddr1;
sendbuf[tx_len++] = srcaddr2;
sendbuf[tx_len++] = srcaddr3;
sendbuf[tx_len++] = dstaddr0;
sendbuf[tx_len++] = dstaddr1;
sendbuf[tx_len++] = dstaddr2;
sendbuf[tx_len++] = dstaddr3;
sendbuf[tx_len++] = srcport1;
sendbuf[tx_len++] = srcport2;
sendbuf[tx_len++] = dstport1;
      sendbuf[tx_len++] = dstport2;
      sendbuf[tx_len++] = lengt1;
      sendbuf[tx_len++] = lengt2;
      sendbuf[tx_len++] = chksum1;
      sendbuf[tx_len++] = chksum2;

sendbuf[tx_len++] = len_id;
      sendbuf[tx_len++] = t_id;

/*  sendbuf[tx_len++] = session_id;
      sendbuf[tx_len++] = length_id;
      sendbuf[tx_len++] = common_header;
      sendbuf[tx_len++] = l_id;
      sendbuf[tx_len++] = spdu_length1;
      sendbuf[tx_len++] = spdu_length2;
      sendbuf[tx_len++] = spdu_length3;
      sendbuf[tx_len++] = spdu_length4;
sendbuf[tx_len++] = spdu_num1;
      sendbuf[tx_len++] = spdu_num2;
      sendbuf[tx_len++] = spdu_num3;
```

```
	sendbuf[tx_len++] = spdu_num4;
	sendbuf[tx_len++] = ver1;
	sendbuf[tx_len++] = ver2;
sendbuf[tx_len++] = TimeofCurrentKey1;
sendbuf[tx_len++] = TimeofCurrentKey2;
sendbuf[tx_len++] = TimeofCurrentKey3;
	sendbuf[tx_len++] = TimeofCurrentKey4;
sendbuf[tx_len++] = TimeofNextKey1;
sendbuf[tx_len++] = TimeofNextKey2;
	sendbuf[tx_len++] = sa1;
	sendbuf[tx_len++] = sa2;
	sendbuf[tx_len++] = keyID1;
	sendbuf[tx_len++] = keyID2;
	sendbuf[tx_len++] = keyID3;
	sendbuf[tx_len++] = keyID4;
sendbuf[tx_len++] = len1;
	sendbuf[tx_len++] = len2;
	sendbuf[tx_len++] = len3;
	sendbuf[tx_len++] = len4;
	sendbuf[tx_len++] = pl_type;
	sendbuf[tx_len++] = simulation;
	sendbuf[tx_len++] = APPID1;
	sendbuf[tx_len++] = APPID2;
	sendbuf[tx_len++] = length1;
	sendbuf[tx_len++] = length2;

	for(j=0;j<137;j++)
	    sendbuf[tx_len++] = goosedata[j];
	*/

	for(j=0;j<173;j++)
	    sendbuf[tx_len++] = ciphertext[j];
	sendbuf[tx_len++] = signature;
	sendbuf[tx_len++] = sig_len;

/*
	for(j=0;j<32;j++)
	    sendbuf[tx_len++] = hash[j]; */
	for(j=0;j<32;j++)
	    sendbuf[tx_len++] = sig_val[j];

	/*sendbuf[tx_len++] = 0x3C;
	sendbuf[tx_len++] = 0x1D;
	sendbuf[tx_len++] = 0x1C;
	sendbuf[tx_len++] = 0x98;  */
```

```
            sendbuf[tx_len++] = 0xEF;
            sendbuf[tx_len++] = 0x1B;
            sendbuf[tx_len++] = 0x13;
            sendbuf[tx_len++] = 0x81;

            /* Index of the network device */
        socket_address.sll_ifindex = if_idx.ifr_ifindex;  /* Network Interface number */

        /* Address length*/
        socket_address.sll_halen = ETH_ALEN; /* Length of Ethernet address */

        /* Destination MAC */
        socket_address.sll_addr[0] = DEST_MAC0;
        socket_address.sll_addr[1] = DEST_MAC1;
        socket_address.sll_addr[2] = DEST_MAC2;
        socket_address.sll_addr[3] = DEST_MAC3;
        socket_address.sll_addr[4] = DEST_MAC4;
        socket_address.sll_addr[5] = DEST_MAC5;

        /* Send packet */
        if (sendto(sfd, sendbuf, tx_len, 0, (struct sockaddr*)&socket_address, sizeof(struct sockaddr_ll)) < 0)
            printf("Send failed\n");
        else {
            printf("Sent :");
            for (i=0; i < tx_len; i++)
                printf("%02x:", sendbuf[i]);
            printf("\n");
        }
        /* Wait specified number of microseconds
           1,000,000 microseconds = 1 second
           */
        usleep(1000000);
    }
    return 0;
}
```

APPENDIX E

EtM Sender Source Code

```
/*This file is part of Github*/
/*https://github.com/61850security/S-GoSV-part-2*/
/* Source code published by (Hussain, Farooq, et al., 2020) */
/*This program is published*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <openssl/hmac.h>
#include <openssl/evp.h>
#include <sys/time.h>
#include <arpa/inet.h>
#include <linux/if_packet.h>
#include <string.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <net/if.h>
#include <netinet/ether.h>
#include <unistd.h>

/*
#define DEST_MAC0    0xFF
#define DEST_MAC1    0xFF
#define DEST_MAC2    0xFF
#define DEST_MAC3    0xFF
#define DEST_MAC4    0xFF
#define DEST_MAC5    0xFF
*/

#define DEST_MAC0    0x08
#define DEST_MAC1    0x00
#define DEST_MAC2    0x27
#define DEST_MAC3    0xF8
#define DEST_MAC4    0x42
#define DEST_MAC5    0xA7

// We can set standard multicast destination address as 01 0C CD 01 03 FF

#define IF_NAME     "eth0"  /* your system interface name */
#define BUF_SIZ     2048
```

```
/* Session layer fields */

char APPID1=0x00;
char APPID2=0x01;
char length1=0x00;
char length2=0x66;
char resrv1_1=0x00;                    /* int16_t resrv1;  */
char resrv1_2=0x00;
char resrv2_1=0x00;                    /* int16_t resrv2 */
char resrv2_2=0x00;
char session_id = 0xA2;
char length_id = 0x18;
char common_header= 0x80;
char l_id = 0x16;
char spdu_length1=0x00;
char spdu_length2=0x00;
char spdu_length3=0x00;
char spdu_length4=0xA4;
char spdu_num1=0x00;
char spdu_num2=0x00;
char spdu_num3=0x00;
char spdu_num4=0x0C;
char ver1=0x00;
char ver2=0x01;
char TimeofCurrentKey1=0x5B;
char TimeofCurrentKey2=0xFC;
char TimeofCurrentKey3=0xF6;
char TimeofCurrentKey4=0xB0;
char TimeofNextKey1=0x00;
char TimeofNextKey2=0x3C;
char sa1=0x02;
char sa2=0x03;
char keyID1=0x00;
char keyID2=0x00;
char keyID3=0x00;
char keyID4=0x0D;
char len1=0x00;
char len2=0x00;
char len3=0x00;
char len4=0x6C;
char pl_type=0x82;
char simulation=0x01;
```

```
/* Sample Value fields according to IEC 61850-9-2 */
char sav_PDU_tag=0x60;              /* sav_PDU tag  */
char sav_PDU_length=0x64;               /* sav_PDU length - size of APDU */
char noASDU_tag=0x80;                   /* number of ASDU tag  */
char noASDU_length=0x01;            /* number of ASDU length */
char noASDU=0x01;                  /* noASDU value  */
char SequenceofASDU_tag=0xA2;        /* SequenceofASDU tag   */
char SequenceofASDU_length=0x5F;     /* SequenceofASDU length - size of all ASDU  */
char ASDU_tag=0x30;                 /* ASDU tag */
char ASDU_length=0x5D;              /* ASDU length */
char svID_tag =0x80;                /* Sample Value identifier tag   */
char svID_length =0x0C;             /* Sample Value identifier length */
char svID_1=0x46;                  /*  svID[12] naming   */
char svID_2=0x52;
char svID_3=0x45;
char svID_4=0x41;
char svID_5=0x2D;
char svID_6=0x47;
char svID_7=0x6F;
char svID_8=0x53;
char svID_9=0x56;
char svID_10=0x2D;
char svID_11=0x31;
char svID_12=0x20;
char smpCnt_tag=0x82;               /* sample count tag */
char smpCnt_length =0x02;           /* sample count length */
char smpCnt_1=0x00;                 /* sample count */
char smpCnt_2=0x08;
char confRev_tag=0x83;              /* confRev tag - configuratin revision number */
char confRev_length=0x04;           /* confRev length */
char confRev1=0x00;                 /* confRev value */
char confRev2=0x00;
char confRev3=0x00;
char confRev4=0x01;
char smpSynch_tag = 0x85;           /* smpSynch tag -synchronisation identifier */
char smpSynch_length =0x01;         /* smpSynch_length */
char smpSynch =0x00;                    /* smpSynch value  */
char SequenceofData_tag =0x87;      /* SequenceofData tag */
char SequenceofData_length=0x40;    /* SequenceofData length */


/* Enter your custom measurement values in hexa decimal    */
char a[64]=    { 0x00, 0x00, 0x00, 0x5A, 0x12, 0x15, 0x12, 0x64,
               0x11, 0x12, 0x18, 0x22, 0x14, 0x12, 0x17, 0x16,
                      0x30, 0x42, 0x10, 0x14, 0x12, 0x15, 0x12, 0x64,
		0x11, 0x12, 0x18, 0x22, 0x14, 0x12, 0x17, 0x16,
			0x30, 0x42, 0x10, 0x14, 0x12, 0x15, 0x12, 0x64,
		0x11, 0x12, 0x18, 0x22, 0x14, 0x12, 0x17, 0x16,
			0x30, 0x42, 0x10, 0x14, 0x12, 0x15, 0x12, 0x64,
		0x11, 0x12, 0x18, 0x22, 0x14, 0x12, 0x17, 0x16 };

unsigned char s[256] =
 {
   0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30, 0x01, 0x67, 0x2B, 0xFE, 0xD7, 0xAB, 0x76,
   0xCA, 0x82, 0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0, 0xAD, 0xD4, 0xA2, 0xAF, 0x9C, 0xA4, 0x72, 0xC0,
   0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC, 0x34, 0xA5, 0xE5, 0xF1, 0x71, 0xD8, 0x31, 0x15,
   0x04, 0xC7, 0x23, 0xC3, 0x18, 0x96, 0x05, 0x9A, 0x07, 0x12, 0x80, 0xE2, 0xEB, 0x27, 0xB2, 0x75,
   0x09, 0x83, 0x2C, 0x1A, 0x1B, 0x6E, 0x5A, 0xA0, 0x52, 0x3B, 0xD6, 0xB3, 0x29, 0xE3, 0x2F, 0x84,
   0x53, 0xD1, 0x00, 0xED, 0x20, 0xFC, 0xB1, 0x5B, 0x6A, 0xCB, 0xBE, 0x39, 0x4A, 0x4C, 0x58, 0xCF,
   0xD0, 0xEF, 0xAA, 0xFB, 0x43, 0x4D, 0x33, 0x85, 0x45, 0xF9, 0x02, 0x7F, 0x50, 0x3C, 0x9F, 0xA8,
   0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D, 0x38, 0xF5, 0xBC, 0xB6, 0xDA, 0x21, 0x10, 0xFF, 0xF3, 0xD2,
   0xCD, 0x0C, 0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17, 0xC4, 0xA7, 0x7E, 0x3D, 0x64, 0x5D, 0x19, 0x73,
   0x60, 0x81, 0x4F, 0xDC, 0x22, 0x2A, 0x90, 0x88, 0x46, 0xEE, 0xB8, 0x14, 0xDE, 0x5E, 0x0B, 0xDB,
   0xE0, 0x32, 0x3A, 0x0A, 0x49, 0x06, 0x24, 0x5C, 0xC2, 0xD3, 0xAC, 0x62, 0x91, 0x95, 0xE4, 0x79,
   0xE7, 0xC8, 0x37, 0x6D, 0x8D, 0xD5, 0x4E, 0xA9, 0x6C, 0x56, 0xF4, 0xEA, 0x65, 0x7A, 0xAE, 0x08,
   0xBA, 0x78, 0x25, 0x2E, 0x1C, 0xA6, 0xB4, 0xC6, 0xE8, 0xDD, 0x74, 0x1F, 0x4B, 0xBD, 0x8B, 0x8A,
   0x70, 0x3E, 0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E, 0x61, 0x35, 0x57, 0xB9, 0x86, 0xC1, 0x1D, 0x9E,
   0xE1, 0xF8, 0x98, 0x11, 0x69, 0xD9, 0x8E, 0x94, 0x9B, 0x1E, 0x87, 0xE9, 0xCE, 0x55, 0x28, 0xDF,
   0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68, 0x41, 0x99, 0x2D, 0x0F, 0xB0, 0x54, 0xBB, 0x16
 };
 unsigned char inv_s[256] =
 {
   0x52, 0x09, 0x6A, 0xD5, 0x30, 0x36, 0xA5, 0x38, 0xBF, 0x40, 0xA3, 0x9E, 0x81, 0xF3, 0xD7, 0xFB,
   0x7C, 0xE3, 0x39, 0x82, 0x9B, 0x2F, 0xFF, 0x87, 0x34, 0x8E, 0x43, 0x44, 0xC4, 0xDE, 0xE9, 0xCB,
   0x54, 0x7B, 0x94, 0x32, 0xA6, 0xC2, 0x23, 0x3D, 0xEE, 0x4C, 0x95, 0x0B, 0x42, 0xFA, 0xC3, 0x4E,
   0x08, 0x2E, 0xA1, 0x66, 0x28, 0xD9, 0x24, 0xB2, 0x76, 0x5B, 0xA2, 0x49, 0x6D, 0x8B, 0xD1, 0x25,
   0x72, 0xF8, 0xF6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xD4, 0xA4, 0x5C, 0xCC, 0x5D, 0x65, 0xB6, 0x92,
   0x6C, 0x70, 0x48, 0x50, 0xFD, 0xED, 0xB9, 0xDA, 0x5E, 0x15, 0x46, 0x57, 0xA7, 0x8D, 0x9D, 0x84,
   0x90, 0xD8, 0xAB, 0x00, 0x8C, 0xBC, 0xD3, 0x0A, 0xF7, 0xE4, 0x58, 0x05, 0xB8, 0xB3, 0x45, 0x06,
   0xD0, 0x2C, 0x1E, 0x8F, 0xCA, 0x3F, 0x0F, 0x02, 0xC1, 0xAF, 0xBD, 0x03, 0x01, 0x13, 0x8A, 0x6B,
   0x3A, 0x91, 0x11, 0x41, 0x4F, 0x67, 0xDC, 0xEA, 0x97, 0xF2, 0xCF, 0xCE, 0xF0, 0xB4, 0xE6, 0x73,
   0x96, 0xAC, 0x74, 0x22, 0xE7, 0xAD, 0x35, 0x85, 0xE2, 0xF9, 0x37, 0xE8, 0x1C, 0x75, 0xDF, 0x6E,
   0x47, 0xF1, 0x1A, 0x71, 0x1D, 0x29, 0xC5, 0x89, 0x6F, 0xB7, 0x62, 0x0E, 0xAA, 0x18, 0xBE, 0x1B,
   0xFC, 0x56, 0x3E, 0x4B, 0xC6, 0xD2, 0x79, 0x20, 0x9A, 0xDB, 0xC0, 0xFE, 0x78, 0xCD, 0x5A, 0xF4,
   0x1F, 0xDD, 0xA8, 0x33, 0x88, 0x07, 0xC7, 0x31, 0xB1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xEC, 0x5F,
   0x60, 0x51, 0x7F, 0xA9, 0x19, 0xB5, 0x4A, 0x0D, 0x2D, 0xE5, 0x7A, 0x9F, 0x93, 0xC9, 0x9C, 0xEF,
   0xA0, 0xE0, 0x3B, 0x4D, 0xAE, 0x2A, 0xF5, 0xB0, 0xC8, 0xEB, 0xBB, 0x3C, 0x83, 0x53, 0x99, 0x61,
   0x17, 0x2B, 0x04, 0x7E, 0xBA, 0x77, 0xD6, 0x26, 0xE1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0C, 0x7D
 };
```

```
unsigned char mul2[] =
{
    0x00,0x02,0x04,0x06,0x08,0x0a,0x0c,0x0e,0x10,0x12,0x14,0x16,0x18,0x1a,0x1c,0x1e,
    0x20,0x22,0x24,0x26,0x28,0x2a,0x2c,0x2e,0x30,0x32,0x34,0x36,0x38,0x3a,0x3c,0x3e,
    0x40,0x42,0x44,0x46,0x48,0x4a,0x4c,0x4e,0x50,0x52,0x54,0x56,0x58,0x5a,0x5c,0x5e,
    0x60,0x62,0x64,0x66,0x68,0x6a,0x6c,0x6e,0x70,0x72,0x74,0x76,0x78,0x7a,0x7c,0x7e,
    0x80,0x82,0x84,0x86,0x88,0x8a,0x8c,0x8e,0x90,0x92,0x94,0x96,0x98,0x9a,0x9c,0x9e,
    0xa0,0xa2,0xa4,0xa6,0xa8,0xaa,0xac,0xae,0xb0,0xb2,0xb4,0xb6,0xb8,0xba,0xbc,0xbe,
    0xc0,0xc2,0xc4,0xc6,0xc8,0xca,0xcc,0xce,0xd0,0xd2,0xd4,0xd6,0xd8,0xda,0xdc,0xde,
    0xe0,0xe2,0xe4,0xe6,0xe8,0xea,0xec,0xee,0xf0,0xf2,0xf4,0xf6,0xf8,0xfa,0xfc,0xfe,
    0x1b,0x19,0x1f,0x1d,0x13,0x11,0x17,0x15,0x0b,0x09,0x0f,0x0d,0x03,0x01,0x07,0x05,
    0x3b,0x39,0x3f,0x3d,0x33,0x31,0x37,0x35,0x2b,0x29,0x2f,0x2d,0x23,0x21,0x27,0x25,
    0x5b,0x59,0x5f,0x5d,0x53,0x51,0x57,0x55,0x4b,0x49,0x4f,0x4d,0x43,0x41,0x47,0x45,
    0x7b,0x79,0x7f,0x7d,0x73,0x71,0x77,0x75,0x6b,0x69,0x6f,0x6d,0x63,0x61,0x67,0x65,
    0x9b,0x99,0x9f,0x9d,0x93,0x91,0x97,0x95,0x8b,0x89,0x8f,0x8d,0x83,0x81,0x87,0x85,
    0xbb,0xb9,0xbf,0xbd,0xb3,0xb1,0xb7,0xb5,0xab,0xa9,0xaf,0xad,0xa3,0xa1,0xa7,0xa5,
    0xdb,0xd9,0xdf,0xdd,0xd3,0xd1,0xd7,0xd5,0xcb,0xc9,0xcf,0xcd,0xc3,0xc1,0xc7,0xc5,
    0xfb,0xf9,0xff,0xfd,0xf3,0xf1,0xf7,0xf5,0xeb,0xe9,0xef,0xed,0xe3,0xe1,0xe7,0xe5
};

unsigned char mul_3[] =
{
    0x00,0x03,0x06,0x05,0x0c,0x0f,0x0a,0x09,0x18,0x1b,0x1e,0x1d,0x14,0x17,0x12,0x11,
    0x30,0x33,0x36,0x35,0x3c,0x3f,0x3a,0x39,0x28,0x2b,0x2e,0x2d,0x24,0x27,0x22,0x21,
    0x60,0x63,0x66,0x65,0x6c,0x6f,0x6a,0x69,0x78,0x7b,0x7e,0x7d,0x74,0x77,0x72,0x71,
    0x50,0x53,0x56,0x55,0x5c,0x5f,0x5a,0x59,0x48,0x4b,0x4e,0x4d,0x44,0x47,0x42,0x41,
    0xc0,0xc3,0xc6,0xc5,0xcc,0xcf,0xca,0xc9,0xd8,0xdb,0xde,0xdd,0xd4,0xd7,0xd2,0xd1,
    0xf0,0xf3,0xf6,0xf5,0xfc,0xff,0xfa,0xf9,0xe8,0xeb,0xee,0xed,0xe4,0xe7,0xe2,0xe1,
    0xa0,0xa3,0xa6,0xa5,0xac,0xaf,0xaa,0xa9,0xb8,0xbb,0xbe,0xbd,0xb4,0xb7,0xb2,0xb1,
    0x90,0x93,0x96,0x95,0x9c,0x9f,0x9a,0x99,0x88,0x8b,0x8e,0x8d,0x84,0x87,0x82,0x81,
    0x9b,0x98,0x9d,0x9e,0x97,0x94,0x91,0x92,0x83,0x80,0x85,0x86,0x8f,0x8c,0x89,0x8a,
    0xab,0xa8,0xad,0xae,0xa7,0xa4,0xa1,0xa2,0xb3,0xb0,0xb5,0xb6,0xbf,0xbc,0xb9,0xba,
    0xfb,0xf8,0xfd,0xfe,0xf7,0xf4,0xf1,0xf2,0xe3,0xe0,0xe5,0xe6,0xef,0xec,0xe9,0xea,
    0xcb,0xc8,0xcd,0xce,0xc7,0xc4,0xc1,0xc2,0xd3,0xd0,0xd5,0xd6,0xdf,0xdc,0xd9,0xda,
    0x5b,0x58,0x5d,0x5e,0x57,0x54,0x51,0x52,0x43,0x40,0x45,0x46,0x4f,0x4c,0x49,0x4a,
    0x6b,0x68,0x6d,0x6e,0x67,0x64,0x61,0x62,0x73,0x70,0x75,0x76,0x7f,0x7c,0x79,0x7a,
    0x3b,0x38,0x3d,0x3e,0x37,0x34,0x31,0x32,0x23,0x20,0x25,0x26,0x2f,0x2c,0x29,0x2a,
    0x0b,0x08,0x0d,0x0e,0x07,0x04,0x01,0x02,0x13,0x10,0x15,0x16,0x1f,0x1c,0x19,0x1a
};

unsigned char mul_9[] =
{
    0x00,0x09,0x12,0x1b,0x24,0x2d,0x36,0x3f,0x48,0x41,0x5a,0x53,0x6c,0x65,0x7e,0x77,
    0x90,0x99,0x82,0x8b,0xb4,0xbd,0xa6,0xaf,0xd8,0xd1,0xca,0xc3,0xfc,0xf5,0xee,0xe7,
    0x3b,0x32,0x29,0x20,0x1f,0x16,0x0d,0x04,0x73,0x7a,0x61,0x68,0x57,0x5e,0x45,0x4c,
    0xab,0xa2,0xb9,0xb0,0x8f,0x86,0x9d,0x94,0xe3,0xea,0xf1,0xf8,0xc7,0xce,0xd5,0xdc,
    0x76,0x7f,0x64,0x6d,0x52,0x5b,0x40,0x49,0x3e,0x37,0x2c,0x25,0x1a,0x13,0x08,0x01,
    0xe6,0xef,0xf4,0xfd,0xc2,0xcb,0xd0,0xd9,0xae,0xa7,0xbc,0xb5,0x8a,0x83,0x98,0x91,
    0x4d,0x44,0x5f,0x56,0x69,0x60,0x7b,0x72,0x05,0x0c,0x17,0x1e,0x21,0x28,0x33,0x3a,
    0xdd,0xd4,0xcf,0xc6,0xf9,0xf0,0xeb,0xe2,0x95,0x9c,0x87,0x8e,0xb1,0xb8,0xa3,0xaa,
    0xec,0xe5,0xfe,0xf7,0xc8,0xc1,0xda,0xd3,0xa4,0xad,0xb6,0xbf,0x80,0x89,0x92,0x9b,
    0x7c,0x75,0x6e,0x67,0x58,0x51,0x4a,0x43,0x34,0x3d,0x26,0x2f,0x10,0x19,0x02,0x0b,
    0xd7,0xde,0xc5,0xcc,0xf3,0xfa,0xe1,0xe8,0x9f,0x96,0x8d,0x84,0xbb,0xb2,0xa9,0xa0,
    0x47,0x4e,0x55,0x5c,0x63,0x6a,0x71,0x78,0x0f,0x06,0x1d,0x14,0x2b,0x22,0x39,0x30,
    0x9a,0x93,0x88,0x81,0xbe,0xb7,0xac,0xa5,0xd2,0xdb,0xc0,0xc9,0xf6,0xff,0xe4,0xed,
    0x0a,0x03,0x18,0x11,0x2e,0x27,0x3c,0x35,0x42,0x4b,0x50,0x59,0x66,0x6f,0x74,0x7d,
    0xa1,0xa8,0xb3,0xba,0x85,0x8c,0x97,0x9e,0xe9,0xe0,0xfb,0xf2,0xcd,0xc4,0xdf,0xd6,
    0x31,0x38,0x23,0x2a,0x15,0x1c,0x07,0x0e,0x79,0x70,0x6b,0x62,0x5d,0x54,0x4f,0x46
};

unsigned char mul_11[] =
{
    0x00,0x0b,0x16,0x1d,0x2c,0x27,0x3a,0x31,0x58,0x53,0x4e,0x45,0x74,0x7f,0x62,0x69,
    0xb0,0xbb,0xa6,0xad,0x9c,0x97,0x8a,0x81,0xe8,0xe3,0xfe,0xf5,0xc4,0xcf,0xd2,0xd9,
    0x7b,0x70,0x6d,0x66,0x57,0x5c,0x41,0x4a,0x23,0x28,0x35,0x3e,0x0f,0x04,0x19,0x12,
    0xcb,0xc0,0xdd,0xd6,0xe7,0xec,0xf1,0xfa,0x93,0x98,0x85,0x8e,0xbf,0xb4,0xa9,0xa2,
    0xf6,0xfd,0xe0,0xeb,0xda,0xd1,0xcc,0xc7,0xae,0xa5,0xb8,0xb3,0x82,0x89,0x94,0x9f,
    0x46,0x4d,0x50,0x5b,0x6a,0x61,0x7c,0x77,0x1e,0x15,0x08,0x03,0x32,0x39,0x24,0x2f,
    0x8d,0x86,0x9b,0x90,0xa1,0xaa,0xb7,0xbc,0xd5,0xde,0xc3,0xc8,0xf9,0xf2,0xef,0xe4,
    0x3d,0x36,0x2b,0x20,0x11,0x1a,0x07,0x0c,0x65,0x6e,0x73,0x78,0x49,0x42,0x5f,0x54,
    0xf7,0xfc,0xe1,0xea,0xdb,0xd0,0xcd,0xc6,0xaf,0xa4,0xb9,0xb2,0x83,0x88,0x95,0x9e,
    0x47,0x4c,0x51,0x5a,0x6b,0x60,0x7d,0x76,0x1f,0x14,0x09,0x02,0x33,0x38,0x25,0x2e,
    0x8c,0x87,0x9a,0x91,0xa0,0xab,0xb6,0xbd,0xd4,0xdf,0xc2,0xc9,0xf8,0xf3,0xee,0xe5,
    0x3c,0x37,0x2a,0x21,0x10,0x1b,0x06,0x0d,0x64,0x6f,0x72,0x79,0x48,0x43,0x5e,0x55,
    0x01,0x0a,0x17,0x1c,0x2d,0x26,0x3b,0x30,0x59,0x52,0x4f,0x44,0x75,0x7e,0x63,0x68,
    0xb1,0xba,0xa7,0xac,0x9d,0x96,0x8b,0x80,0xe9,0xe2,0xff,0xf4,0xc5,0xce,0xd3,0xd8,
    0x7a,0x71,0x6c,0x67,0x56,0x5d,0x40,0x4b,0x22,0x29,0x34,0x3f,0x0e,0x05,0x18,0x13,
    0xca,0xc1,0xdc,0xd7,0xe6,0xed,0xf0,0xfb,0x92,0x99,0x84,0x8f,0xbe,0xb5,0xa8,0xa3
};

unsigned char mul_13[] =
{
    0x00,0x0d,0x1a,0x17,0x34,0x39,0x2e,0x23,0x68,0x65,0x72,0x7f,0x5c,0x51,0x46,0x4b,
    0xd0,0xdd,0xca,0xc7,0xe4,0xe9,0xfe,0xf3,0xb8,0xb5,0xa2,0xaf,0x8c,0x81,0x96,0x9b,
    0xbb,0xb6,0xa1,0xac,0x8f,0x82,0x95,0x98,0xd3,0xde,0xc9,0xc4,0xe7,0xea,0xfd,0xf0,
    0x6b,0x66,0x71,0x7c,0x5f,0x52,0x45,0x48,0x03,0x0e,0x19,0x14,0x37,0x3a,0x2d,0x20,
    0x6d,0x60,0x77,0x7a,0x59,0x54,0x43,0x4e,0x05,0x08,0x1f,0x12,0x31,0x3c,0x2b,0x26,
    0xbd,0xb0,0xa7,0xaa,0x89,0x84,0x93,0x9e,0xd5,0xd8,0xcf,0xc2,0xe1,0xec,0xfb,0xf6,
    0xd6,0xdb,0xcc,0xc1,0xe2,0xef,0xf8,0xf5,0xbe,0xb3,0xa4,0xa9,0x8a,0x87,0x90,0x9d,
    0x06,0x0b,0x1c,0x11,0x32,0x3f,0x28,0x25,0x6e,0x63,0x74,0x79,0x5a,0x57,0x40,0x4d,
    0xda,0xd7,0xc0,0xcd,0xee,0xe3,0xf4,0xf9,0xb2,0xbf,0xa8,0xa5,0x86,0x8b,0x9c,0x91,
    0x0a,0x07,0x10,0x1d,0x3e,0x33,0x24,0x29,0x62,0x6f,0x78,0x75,0x56,0x5b,0x4c,0x41,
    0x61,0x6c,0x7b,0x76,0x55,0x58,0x4f,0x42,0x09,0x04,0x13,0x1e,0x3d,0x30,0x27,0x2a,
    0xb1,0xbc,0xab,0xa6,0x85,0x88,0x9f,0x92,0xd9,0xd4,0xc3,0xce,0xed,0xe0,0xf7,0xfa,
    0xb7,0xba,0xad,0xa0,0x83,0x8e,0x99,0x94,0xdf,0xd2,0xc5,0xc8,0xeb,0xe6,0xf1,0xfc,
    0x67,0x6a,0x7d,0x70,0x53,0x5e,0x49,0x44,0x0f,0x02,0x15,0x18,0x3b,0x36,0x21,0x2c,
    0x0c,0x01,0x16,0x1b,0x38,0x35,0x22,0x2f,0x64,0x69,0x7e,0x73,0x50,0x5d,0x4a,0x47,
    0xdc,0xd1,0xc6,0xcb,0xe8,0xe5,0xf2,0xff,0xb4,0xb9,0xae,0xa3,0x80,0x8d,0x9a,0x97
};

unsigned char mul_14[] =
{
    0x00,0x0e,0x1c,0x12,0x38,0x36,0x24,0x2a,0x70,0x7e,0x6c,0x62,0x48,0x46,0x54,0x5a,
    0xe0,0xee,0xfc,0xf2,0xd8,0xd6,0xc4,0xca,0x90,0x9e,0x8c,0x82,0xa8,0xa6,0xb4,0xba,
    0xdb,0xd5,0xc7,0xc9,0xe3,0xed,0xff,0xf1,0xab,0xa5,0xb7,0xb9,0x93,0x9d,0x8f,0x81,
    0x3b,0x35,0x27,0x29,0x03,0x0d,0x1f,0x11,0x4b,0x45,0x57,0x59,0x73,0x7d,0x6f,0x61,
    0xad,0xa3,0xb1,0xbf,0x95,0x9b,0x89,0x87,0xdd,0xd3,0xc1,0xcf,0xe5,0xeb,0xf9,0xf7,
    0x4d,0x43,0x51,0x5f,0x75,0x7b,0x69,0x67,0x3d,0x33,0x21,0x2f,0x05,0x0b,0x19,0x17,
    0x76,0x78,0x6a,0x64,0x4e,0x40,0x52,0x5c,0x06,0x08,0x1a,0x14,0x3e,0x30,0x22,0x2c,
    0x96,0x98,0x8a,0x84,0xae,0xa0,0xb2,0xbc,0xe6,0xe8,0xfa,0xf4,0xde,0xd0,0xc2,0xcc,
    0x41,0x4f,0x5d,0x53,0x79,0x77,0x65,0x6b,0x31,0x3f,0x2d,0x23,0x09,0x07,0x15,0x1b,
    0xa1,0xaf,0xbd,0xb3,0x99,0x97,0x85,0x8b,0xd1,0xdf,0xcd,0xc3,0xe9,0xe7,0xf5,0xfb,
    0x9a,0x94,0x86,0x88,0xa2,0xac,0xbe,0xb0,0xea,0xe4,0xf6,0xf8,0xd2,0xdc,0xce,0xc0,
    0x7a,0x74,0x66,0x68,0x42,0x4c,0x5e,0x50,0x0a,0x04,0x16,0x18,0x32,0x3c,0x2e,0x20,
    0xec,0xe2,0xf0,0xfe,0xd4,0xda,0xc8,0xc6,0x9c,0x92,0x80,0x8e,0xa4,0xaa,0xb8,0xb6,
    0x0c,0x02,0x10,0x1e,0x34,0x3a,0x28,0x26,0x7c,0x72,0x60,0x6e,0x44,0x4a,0x58,0x56,
    0x37,0x39,0x2b,0x25,0x0f,0x01,0x13,0x1d,0x47,0x49,0x5b,0x55,0x7f,0x71,0x63,0x6d,
    0xd7,0xd9,0xcb,0xc5,0xef,0xe1,0xf3,0xfd,0xa7,0xa9,0xbb,0xb5,0x9f,0x91,0x83,0x8d
};

unsigned char rcon[11] =
{
    0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36,
};

unsigned char * g (unsigned char wInput[4], int counter)
{
    unsigned char * wReady = malloc(4);
    unsigned char temp[4] = "";
    unsigned char a = wInput[0];
    for(int i =0;i<3; i++)
    {
        temp[i] = wInput[(i+1)];
    }
    temp[3] = a;
    for (int i =0; i<4;i++)
        temp[i] = s[temp[i]];

    //unsigned char array formed for xoring with rcon

    unsigned char array2[4] = "";
    array2[0] = rcon[counter];
    array2[1] = array2[2] = array2[3] = 0x00;

    for (int i=0;i<4;i++)
    wReady[i] = temp[i] ^ array2[i];
    return wReady;
}

unsigned char * keyExpansion(unsigned char key[16])
{
    unsigned char words[44][4];
    for (int i = 0; i < 44; ++i)
    {
        for (int j = 0; j <4; ++j)
        {
            words[i][j]=0x00;
        }
    }

    unsigned char * expandedKey = malloc(176);

    int byteCount = 0; //this is to keep a count on the bytes of the expandedKey array

    for (int i=0;i<16;i++)
            expandedKey[i] = key[i];

    for(int j=0;j<4;j++)
    {
         for(int k=0;k<4;k++)
         {
            words[j][k] = expandedKey[byteCount];
            byteCount++;
         }
    }
    for(int l=4;l<44;l++)
    {
        if((l%4)==0)
        {
            for(int m=0;m<4;m++)
            {
                words[l][m] = words[(l-4)][m] ^ g(words[l-1], (l/4))[m];
            }
        }
        else
        {
            for(int n=0;n<4;n++)
            {
                words[l][n] = words[l-1][n] ^ words[l-4][n];
            }
        }
    }

    int loc=0;
    for(int i=0;i<44;i++ )
    {
        for(int j=0;j<4;j++)
        {
            expandedKey[loc] = words[i][j];
            loc++;
        }
    }
    return expandedKey;
}

void mixColumns(unsigned char * plainText)
{
    unsigned char * tempC = malloc(16);

    for (int i = 0; i < 4; ++i)
    {
        tempC[(4*i)+0] = (unsigned char) (mul2[plainText[(4*i)+0]] ^ mul_3[plainText[(4*i)+1]] ^ plainText[(4*i)+2] ^
plainText[(4*i)+3]);
        tempC[(4*i)+1] = (unsigned char) (plainText[(4*i)+0] ^ mul2[plainText[(4*i)+1]] ^ mul_3[plainText[(4*i)+2]] ^
plainText[(4*i)+3]);
        tempC[(4*i)+2] = (unsigned char) (plainText[(4*i)+0] ^ plainText[(4*i)+1] ^ mul2[plainText[(4*i)+2]] ^
mul_3[plainText[(4*i)+3]]);
        tempC[(4*i)+3] = (unsigned char) (mul_3[plainText[(4*i)+0]] ^ plainText[(4*i)+1] ^ plainText[(4*i)+2] ^
mul2[plainText[(4*i)+3]]);
    }

    for (int i = 0; i < 16; ++i)
    {
        plainText[i] = tempC[i];
    }
    free(tempC);
}
void inverseMixedColumn (unsigned char * plainText)
{
    unsigned char * tempC = malloc(18);

    for (int i = 0; i < 4; ++i)
    {
        tempC[(4*i)+0] = (unsigned char) (mul_14[plainText[(4*i)+0]] ^ mul_11[plainText[(4*i)+1]] ^
mul_13[plainText[(4*i)+2]] ^ mul_9[plainText[(4*i)+3]]);
        tempC[(4*i)+1] = (unsigned char) (mul_9[plainText[(4*i)+0]] ^ mul_14[plainText[(4*i)+1]] ^
mul_11[plainText[(4*i)+2]] ^ mul_13[plainText[(4*i)+3]]);
        tempC[(4*i)+2] = (unsigned char) (mul_13[plainText[(4*i)+0]] ^ mul_9[plainText[(4*i)+1]] ^
mul_14[plainText[(4*i)+2]] ^ mul_11[plainText[(4*i)+3]]);
        tempC[(4*i)+3] = (unsigned char) (mul_11[plainText[(4*i)+0]] ^ mul_13[plainText[(4*i)+1]] ^
mul_9[plainText[(4*i)+2]] ^ mul_14[plainText[(4*i)+3]]);
    }
    for (int i = 0; i < 16; ++i)
    {
        plainText[i] = tempC[i];
    }
    free(tempC);
}
void byteSubShiftRow(unsigned char * state)
{

    unsigned char tmp[16];

    tmp[0] = s[state[0]];
    tmp[1] = s[state[5]];
    tmp[2] = s[state[10]];
    tmp[3] = s[state[15]];

    tmp[4] = s[state[4]];
    tmp[5] = s[state[9]];
    tmp[6] = s[state[14]];
    tmp[7] = s[state[3]];

    tmp[8] = s[state[8]];
    tmp[9] = s[state[13]];
    tmp[10] = s[state[2]];
    tmp[11] = s[state[7]];

    tmp[12] = s[state[12]];
    tmp[13] = s[state[1]];
    tmp[14] = s[state[6]];
    tmp[15] = s[state[11]];

    for(int i=0;i<16;i++)
    {
        state[i] = tmp[i];
    }
}
void inverseByteSubShiftRow(unsigned char * plainText)
{
    unsigned char * temp = malloc(16);
    temp[0] = inv_s[plainText[0]];
    temp[1] = inv_s[plainText[13]];
    temp[2] = inv_s[plainText[10]];
    temp[3] = inv_s[plainText[7]];
    temp[4] = inv_s[plainText[4]];
    temp[5] = inv_s[plainText[1]];
    temp[6] = inv_s[plainText[14]];
    temp[7] = inv_s[plainText[11]];
    temp[8] = inv_s[plainText[8]];
    temp[9] = inv_s[plainText[5]];
    temp[10] = inv_s[plainText[2]];
    temp[11] = inv_s[plainText[15]];
    temp[12] = inv_s[plainText[12]];
    temp[13] = inv_s[plainText[9]];
    temp[14] = inv_s[plainText[6]];
    temp[15] = inv_s[plainText[3]];

    for (int i = 0; i < 16; ++i)
        plainText[i] = temp[i];

    free(temp);
}

unsigned char * AESEncryption(unsigned char * plainText, unsigned char * expandedKey)
{

    unsigned char * state = malloc(16);
    unsigned char * cipherText = malloc(16);
    //unsigned char * expandedKey = malloc(176);
    //expandedKey = keyExpansion(Key);
    //key addition for the first round
    for (int i = 0; i < 16; ++i)
    {
     state[i] = plainText[i] ^ expandedKey[i];
    }

    //now the 9 rounds begin
    for(int rounds = 1; rounds<10; rounds++)
    {
        byteSubShiftRow(state);
        mixColumns(state);
        int counter = 0;
        int loc = rounds*16;
        while(counter<16)
        {
            state[counter] ^= expandedKey[loc];
            loc++;
            counter++;
        }
    }

    //10th round
    byteSubShiftRow(state);
    for(int i=0; i<16;i++)
    {
        cipherText[i] = state[i] ^ expandedKey[i+160];
        //printf("\n cipher[%d]=%x",i,cipher[i]);
    }
    free(state);
    return cipherText;

}

int main(int argc, char *argv[])
{
    unsigned char PDU[BUF_SIZ];
    unsigned char InputData[16], Data[16],cipher_str[1024]="";
    unsigned char plainText[16];
    unsigned char key[14]= { 0x32, 0x21, 0x23, 0x52, 0x71, 0x98, 0x24, 0x03, 0x38, 0x27, 0x01, 0x12, 0x95, 0x23};
    unsigned char * expandedKey = malloc(176);
    unsigned char * cipher = malloc(16);
    unsigned char * plainText1 = malloc(16);
    unsigned char *hash;
    unsigned char hash_string[32];
    int inx=0,i=0,j=0,x,y,z=0,n,p=0,q=0,temp,temp1,p_index=0,index,count=1;
    double begin,end,time_spent,time_hmac=0,time_encrypt=0;

    int sfd,len=0;
    unsigned char sendbuf[BUF_SIZ],ifName[IFNAMSIZ];
    struct sockaddr_ll socket_address; /* The sockaddr_ll structure is a device-independent physical-layer
address.*/
    struct ifreq if_idx, if_mac;

     /* Get interface name */
    strcpy(ifName, IF_NAME);

    /* Open RAW socket to send on */
    if ((sfd = socket(AF_PACKET, SOCK_RAW, IPPROTO_RAW)) == -1)
        perror("socket");

    /* Initiazing the ifreq structure to zero */
    memset(&if_idx, 0, sizeof(struct ifreq));

    /* Copying the interface name */
    strncpy(if_idx.ifr_name, ifName, IFNAMSIZ-1);

    /* Get the index of the interface to send on */
    if (ioctl(sfd, SIOCGIFINDEX, &if_idx) < 0)
        perror("SIOCGIFINDEX");

    expandedKey = keyExpansion(key);

    for(;1;)
    {

            PDU[inx++] = sav_PDU_tag;
            PDU[inx++] = sav_PDU_length;
            PDU[inx++] = noASDU_tag;
            PDU[inx++] = noASDU_length;
            PDU[inx++] = noASDU;
            PDU[inx++] = SequenceofASDU_tag;
            PDU[inx++] = SequenceofASDU_length;
            PDU[inx++] = ASDU_tag;
            PDU[inx++] = ASDU_length;
            PDU[inx++] = svID_tag;
            PDU[inx++] = svID_length;
            PDU[inx++] = svID_1;
            PDU[inx++] = svID_2;
            PDU[inx++] = svID_3;
            PDU[inx++] = svID_4;
            PDU[inx++] = svID_5;
            PDU[inx++] = svID_6;
            PDU[inx++] = svID_7;
            PDU[inx++] = svID_8;
            PDU[inx++] = svID_9;
            PDU[inx++] = svID_10;
            PDU[inx++] = svID_11;
            PDU[inx++] = svID_12;
            PDU[inx++] = smpCnt_tag;
            PDU[inx++] = smpCnt_length;
            PDU[inx++] = smpCnt_1;
            PDU[inx++] = smpCnt_2;
            PDU[inx++] = confRev_tag;
            PDU[inx++] = confRev_length;
            PDU[inx++] = confRev1;
            PDU[inx++] = confRev2;
            PDU[inx++] = confRev3;
            PDU[inx++] = confRev4;
            PDU[inx++] = smpSynch_tag;
            PDU[inx++] = smpSynch_length;
            PDU[inx++] = smpSynch;
            PDU[inx++] = SequenceofData_tag;
            PDU[inx++] = SequenceofData_length;

        /* Buffer of BUF_SIZ bytes we'll construct our frame in. First, clear it all to zero. */
        memset(sendbuf, 0, BUF_SIZ);
        len=0;

        /* Construct the Ethernet header */

        /* Destination address */
        sendbuf[len++] = DEST_MAC0;
        sendbuf[len++] = DEST_MAC1;
        sendbuf[len++] = DEST_MAC2;
        sendbuf[len++] = DEST_MAC3;
        sendbuf[len++] = DEST_MAC4;
        sendbuf[len++] = DEST_MAC5;

        /* Create the source */
        sendbuf[len++] = 0xA0;
        sendbuf[len++] = 0xB3;
        sendbuf[len++] = 0xCC;
        sendbuf[len++] = 0xC5;
        sendbuf[len++] = 0x77;
        sendbuf[len++] = 0xA1;

        sendbuf[len++] = 0x81;
```

```
sendbuf[len++] = 0x00;
sendbuf[len++] = 0x80;
sendbuf[len++] = 0x00;

/* Ethertype field Sampled Value protocol*/
sendbuf[len++] = 0x88;
sendbuf[len++] = 0xBA;

    /*  PDU fields */
sendbuf[len++] = APPID1;
sendbuf[len++] = APPID2;
sendbuf[len++] = length1;
sendbuf[len++] = length2;
sendbuf[len++] = resrv1_1;
    sendbuf[len++] = resrv1_2;
    sendbuf[len++] = resrv2_1;
sendbuf[len++] = resrv2_2;

    sendbuf[len++] = sav_PDU_tag;
    sendbuf[len++] = sav_PDU_length;
    sendbuf[len++] = noASDU_tag;
    sendbuf[len++] = noASDU_length;
    sendbuf[len++] = noASDU;
    sendbuf[len++] = SequenceofASDU_tag;
    sendbuf[len++] = SequenceofASDU_length;
    sendbuf[len++] = ASDU_tag;
    sendbuf[len++] = ASDU_length;
    sendbuf[len++] = svID_tag;
    sendbuf[len++] = svID_length;
    sendbuf[len++] = svID_1;
    sendbuf[len++] = svID_2;
    sendbuf[len++] = svID_3;
    sendbuf[len++] = svID_4;
    sendbuf[len++] = svID_5;
    sendbuf[len++] = svID_6;
    sendbuf[len++] = svID_7;
    sendbuf[len++] = svID_8;
    sendbuf[len++] = svID_9;
    sendbuf[len++] = svID_10;
    sendbuf[len++] = svID_11;
    sendbuf[len++] = svID_12;
    sendbuf[len++] = smpCnt_tag;
    sendbuf[len++] = smpCnt_length;
    sendbuf[len++] = smpCnt_1;
    sendbuf[len++] = smpCnt_2;
    sendbuf[len++] = confRev_tag;
    sendbuf[len++] = confRev_length;
    sendbuf[len++] = confRev1;
    sendbuf[len++] = confRev2;
    sendbuf[len++] = confRev3;
    sendbuf[len++] = confRev4;
    sendbuf[len++] = smpSynch_tag;
    sendbuf[len++] = smpSynch_length;
    sendbuf[len++] = smpSynch;
    sendbuf[len++] = SequenceofData_tag;
    sendbuf[len++] = SequenceofData_length;

/* extension fields*/
    sendbuf[len++] = session_id;
    sendbuf[len++] = length_id;
    sendbuf[len++] = common_header;
    sendbuf[len++] = l_id;
    sendbuf[len++] = spdu_length1;
    sendbuf[len++] = spdu_length2;
    sendbuf[len++] = spdu_length3;
    sendbuf[len++] = spdu_length4;
    sendbuf[len++] = spdu_num1;
    sendbuf[len++] = spdu_num2;
    sendbuf[len++] = spdu_num3;
    sendbuf[len++] = spdu_num4;
    sendbuf[len++] = ver1;
    sendbuf[len++] = ver2;
    sendbuf[len++] = TimeofCurrentKey1;
    sendbuf[len++] = TimeofCurrentKey2;
    sendbuf[len++] = TimeofCurrentKey3;
    sendbuf[len++] = TimeofCurrentKey4;
    sendbuf[len++] = TimeofNextKey1;
    sendbuf[len++] = TimeofNextKey2;
    sendbuf[len++] = sa1;
    sendbuf[len++] = sa2;
    sendbuf[len++] = keyID1;
    sendbuf[len++] = keyID2;
    sendbuf[len++] = keyID3;
    sendbuf[len++] = keyID4;
    sendbuf[len++] = len1;
    sendbuf[len++] = len2;
    sendbuf[len++] = len3;
    sendbuf[len++] = len4;
    sendbuf[len++] = pl_type;
    sendbuf[len++] = simulation;
```

```
//printf(" total packet size=%d",inx);
temp= inx%16;
//printf(" temp=%d",temp);
for (i=0; i<abs(temp-16); i++)
                              PDU[inx++]=0x00;
//printf(" inx value=%d",inx);
      /*for( i=0;i< inx;i++)
    printf(" %.2x",PDU[i]);
*/
temp1= (inx/16);
//printf(" temp1=%d",temp1);
      printf("\n Encrypted text:\n");
      for (x=0;x< temp1 ;x++)
{
        //printf(" \n Plain Text of [%d] chunk :\n",x);
      for (y=0;y<16;y++)
        {
                    plainText[y]= PDU[z++];
               //printf(" %02x",plainText[y]);
        }
                    begin = clock();
      cipher=AESEncryption(plainText,expandedKey);
                    end = clock();
                    time_spent= (double)(end - begin) / CLOCKS_PER_SEC;
        time_encrypt = time_encrypt+time_spent;

      //printf(" \n cipher Text of [%d] chunk :\n",x);
      for (n=0; n<16 ; n++)
                    {
                              printf(" %02x", cipher[n]);
                              sprintf( &(Data[n * 2]) , "%02x", cipher[n]);
               sendbuf[len++]=cipher[n];
                    }
                    strcat(cipher_str,Data);
        //printf("\n cipher-str: %s",cipher_str);
                    //printf(" Data-str :%s",Data);


      }
      /*printf("\n send buf data:\n");
for ( i=0; i<144; i++)
{
                     printf(" %02x",sendbuf[i]);
      } */
    //printf("\n Cipher Text string :\n %s", cipher_str);
begin = clock();
hash = HMAC(EVP_sha256(), key, strlen((char *)key), cipher_str, strlen((char *)cipher_str), NULL, NULL);
    end = clock();
    time_hmac= (double)(end - begin) / CLOCKS_PER_SEC;

begin = clock();
hash = HMAC(EVP_sha256(), key, strlen((char *)key), cipher_str, strlen((char *)cipher_str), NULL, NULL);
    end = clock();
    time_hmac= (double)(end - begin) / CLOCKS_PER_SEC;

printf("\n hash value:");
for (i = 0; i < 32 ; i++)
    {
    sprintf(&(hash_string[i * 2]), "%02x", hash[i]);
      printf("%02x",hash[i]);
                  sendbuf[len++]=hash[i];
    }

/*printf("\n send buf data:\n");
for ( i=0; i<208; i++)
{
                   printf(" %02x",sendbuf[i]);
    } */
//printf("\nHash value: %s", hash_string);
//printf("\n\n");
//printf(" time  =%lf",time_spent);
inx=0;z=0;

    strcpy(cipher_str,"\0");
    //printf("cipher_str=%s",cipher_str);
    //printf("\nencryption time=%lf\n hash generation time=%lf\n",et[index]*1000,ht[index]*1000);

/* Index of the network device */
socket_address.sll_ifindex = if_idx.ifr_ifindex;  /* Network Interface number */

/* Address length*/
socket_address.sll_halen = ETH_ALEN; /* Length of Ethernet address */

/* Destination MAC */
socket_address.sll_addr[0] = DEST_MAC0;
socket_address.sll_addr[1] = DEST_MAC1;
```

```
        socket_address.sll_addr[2] = DEST_MAC2;
        socket_address.sll_addr[3] = DEST_MAC3;
        socket_address.sll_addr[4] = DEST_MAC4;
        socket_address.sll_addr[5] = DEST_MAC5;

        /* Send packet */
        if (sendto(sfd, sendbuf, len, 0, (struct sockaddr*)&socket_address, sizeof(struct sockaddr_ll)) < 0)
            printf("Send failed\n");
        else
        {
            printf("\n packet data Sent :");
            for (i=0; i < len; i++)
                printf("%02x:", sendbuf[i]);
            printf("\n\n\n");

            //printf("\n packet=%d",count++);
            printf("\n MAC generation time=%lf",time_hmac*1000);
            printf("\n Encryption time=%lf\n",time_encrypt*1000);
            printf("\n-------------------------------------\n\n");
            time_hmac=0;time_encrypt=0;
        }
     /* Wait specified number of microseconds 1,000,000 microseconds = 1 second */
        usleep(1000000);
     }

    return 0;
}
```

APPENDIX F

EtM Receiver Source Code

```
/*This file is part of Github*/
/*https://github.com/61850security/S-GoSV-part-2*/
/* Source code published by (Hussain, Farooq, et al., 2020) */
/*This program is published*/

#include <arpa/inet.h>
#include <linux/if_packet.h>
#include <linux/ip.h>
#include <linux/udp.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <net/if.h>
#include <netinet/ether.h>
#include <unistd.h>
#include <math.h>
#include <openssl/hmac.h>
#include <openssl/evp.h>
#include <sys/time.h>

#define DEST_MAC0       0xFF
#define DEST_MAC1       0xFF
#define DEST_MAC2       0xFF
#define DEST_MAC3       0xFF
#define DEST_MAC4       0xFF
#define DEST_MAC5       0xFF

#define ETHER_TYPE      0x88BF

#define DEFAULT_IF      "eth0"  /* your system interface name */
#define BUF_SIZ         2048

unsigned char s[256] =
 {
   0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30, 0x01, 0x67, 0x2B, 0xFE, 0xD7, 0xAB, 0x76,
   0xCA, 0x82, 0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0, 0xAD, 0xD4, 0xA2, 0xAF, 0x9C, 0xA4, 0x72, 0xC0,
   0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC, 0x34, 0xA5, 0xE5, 0xF1, 0x71, 0xD8, 0x31, 0x15,
   0x04, 0xC7, 0x23, 0xC3, 0x18, 0x96, 0x05, 0x9A, 0x07, 0x12, 0x80, 0xE2, 0xEB, 0x27, 0xB2, 0x75,
   0x09, 0x83, 0x2C, 0x1A, 0x1B, 0x6E, 0x5A, 0xA0, 0x52, 0x3B, 0xD6, 0xB3, 0x29, 0xE3, 0x2F, 0x84,
   0x53, 0xD1, 0x00, 0xED, 0x20, 0xFC, 0xB1, 0x5B, 0x6A, 0xCB, 0xBE, 0x39, 0x4A, 0x4C, 0x58, 0xCF,
   0xD0, 0xEF, 0xAA, 0xFB, 0x43, 0x4D, 0x33, 0x85, 0x45, 0xF9, 0x02, 0x7F, 0x50, 0x3C, 0x9F, 0xA8,
   0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D, 0x38, 0xF5, 0xBC, 0xB6, 0xDA, 0x21, 0x10, 0xFF, 0xF3, 0xD2,
   0xCD, 0x0C, 0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17, 0xC4, 0xA7, 0x7E, 0x3D, 0x64, 0x5D, 0x19, 0x73,
   0x60, 0x81, 0x4F, 0xDC, 0x22, 0x2A, 0x90, 0x88, 0x46, 0xEE, 0xB8, 0x14, 0xDE, 0x5E, 0x0B, 0xDB,
   0xE0, 0x32, 0x3A, 0x0A, 0x49, 0x06, 0x24, 0x5C, 0xC2, 0xD3, 0xAC, 0x62, 0x91, 0x95, 0xE4, 0x79,
   0xE7, 0xC8, 0x37, 0x6D, 0x8D, 0xD5, 0x4E, 0xA9, 0x6C, 0x56, 0xF4, 0xEA, 0x65, 0x7A, 0xAE, 0x08,
   0xBA, 0x78, 0x25, 0x2E, 0x1C, 0xA6, 0xB4, 0xC6, 0xE8, 0xDD, 0x74, 0x1F, 0x4B, 0xBD, 0x8B, 0x8A,
   0x70, 0x3E, 0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E, 0x61, 0x35, 0x57, 0xB9, 0x86, 0xC1, 0x1D, 0x9E,
   0xE1, 0xF8, 0x98, 0x11, 0x69, 0xD9, 0x8E, 0x94, 0x9B, 0x1E, 0x87, 0xE9, 0xCE, 0x55, 0x28, 0xDF,
   0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68, 0x41, 0x99, 0x2D, 0x0F, 0xB0, 0x54, 0xBB, 0x16
};

unsigned char inv_s[256] =
{
   0x52, 0x09, 0x6A, 0xD5, 0x30, 0x36, 0xA5, 0x38, 0xBF, 0x40, 0xA3, 0x9E, 0x81, 0xF3, 0xD7, 0xFB,
   0x7C, 0xE3, 0x39, 0x82, 0x9B, 0x2F, 0xFF, 0x87, 0x34, 0x8E, 0x43, 0x44, 0xC4, 0xDE, 0xE9, 0xCB,
   0x54, 0x7B, 0x94, 0x32, 0xA6, 0xC2, 0x23, 0x3D, 0xEE, 0x4C, 0x95, 0x0B, 0x42, 0xFA, 0xC3, 0x4E,
   0x08, 0x2E, 0xA1, 0x66, 0x28, 0xD9, 0x24, 0xB2, 0x76, 0x5B, 0xA2, 0x49, 0x6D, 0x8B, 0xD1, 0x25,
   0x72, 0xF8, 0xF6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xD4, 0xA4, 0x5C, 0xCC, 0x5D, 0x65, 0xB6, 0x92,
   0x6C, 0x70, 0x48, 0x50, 0xFD, 0xED, 0xB9, 0xDA, 0x5E, 0x15, 0x46, 0x57, 0xA7, 0x8D, 0x9D, 0x84,
   0x90, 0xD8, 0xAB, 0x00, 0x8C, 0xBC, 0xD3, 0x0A, 0xF7, 0xE4, 0x58, 0x05, 0xB8, 0xB3, 0x45, 0x06,
   0xD0, 0x2C, 0x1E, 0x8F, 0xCA, 0x3F, 0x0F, 0x02, 0xC1, 0xAF, 0xBD, 0x03, 0x01, 0x13, 0x8A, 0x6B,
   0x3A, 0x91, 0x11, 0x41, 0x4F, 0x67, 0xDC, 0xEA, 0x97, 0xF2, 0xCF, 0xCE, 0xF0, 0xB4, 0xE6, 0x73,
   0x96, 0xAC, 0x74, 0x22, 0xE7, 0xAD, 0x35, 0x85, 0xE2, 0xF9, 0x37, 0xE8, 0x1C, 0x75, 0xDF, 0x6E,
   0x47, 0xF1, 0x1A, 0x71, 0x1D, 0x29, 0xC5, 0x89, 0x6F, 0xB7, 0x62, 0x0E, 0xAA, 0x18, 0xBE, 0x1B,
   0xFC, 0x56, 0x3E, 0x4B, 0xC6, 0xD2, 0x79, 0x20, 0x9A, 0xDB, 0xC0, 0xFE, 0x78, 0xCD, 0x5A, 0xF4,
   0x1F, 0xDD, 0xA8, 0x33, 0x88, 0x07, 0xC7, 0x31, 0xB1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xEC, 0x5F,
   0x60, 0x51, 0x7F, 0xA9, 0x19, 0xB5, 0x4A, 0x0D, 0x2D, 0xE5, 0x7A, 0x9F, 0x93, 0xC9, 0x9C, 0xEF,
   0xA0, 0xE0, 0x3B, 0x4D, 0xAE, 0x2A, 0xF5, 0xB0, 0xC8, 0xEB, 0xBB, 0x3C, 0x83, 0x53, 0x99, 0x61,
   0x17, 0x2B, 0x04, 0x7E, 0xBA, 0x77, 0xD6, 0x26, 0xE1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0C, 0x7D
};

unsigned char mul2[] =
{
   0x00,0x02,0x04,0x06,0x08,0x0a,0x0c,0x0e,0x10,0x12,0x14,0x16,0x18,0x1a,0x1c,0x1e,
   0x20,0x22,0x24,0x26,0x28,0x2a,0x2c,0x2e,0x30,0x32,0x34,0x36,0x38,0x3a,0x3c,0x3e,
   0x40,0x42,0x44,0x46,0x48,0x4a,0x4c,0x4e,0x50,0x52,0x54,0x56,0x58,0x5a,0x5c,0x5e,
   0x60,0x62,0x64,0x66,0x68,0x6a,0x6c,0x6e,0x70,0x72,0x74,0x76,0x78,0x7a,0x7c,0x7e,
   0x80,0x82,0x84,0x86,0x88,0x8a,0x8c,0x8e,0x90,0x92,0x94,0x96,0x98,0x9a,0x9c,0x9e,
   0xa0,0xa2,0xa4,0xa6,0xa8,0xaa,0xac,0xae,0xb0,0xb2,0xb4,0xb6,0xb8,0xba,0xbc,0xbe,
   0xc0,0xc2,0xc4,0xc6,0xc8,0xca,0xcc,0xce,0xd0,0xd2,0xd4,0xd6,0xd8,0xda,0xdc,0xde,
   0xe0,0xe2,0xe4,0xe6,0xe8,0xea,0xec,0xee,0xf0,0xf2,0xf4,0xf6,0xf8,0xfa,0xfc,0xfe,
   0x1b,0x19,0x1f,0x1d,0x13,0x11,0x17,0x15,0x0b,0x09,0x0f,0x0d,0x03,0x01,0x07,0x05,
   0x3b,0x39,0x3f,0x3d,0x33,0x31,0x37,0x35,0x2b,0x29,0x2f,0x2d,0x23,0x21,0x27,0x25,
   0x5b,0x59,0x5f,0x5d,0x53,0x51,0x57,0x55,0x4b,0x49,0x4f,0x4d,0x43,0x41,0x47,0x45,
   0x7b,0x79,0x7f,0x7d,0x73,0x71,0x77,0x75,0x6b,0x69,0x6f,0x6d,0x63,0x61,0x67,0x65,
   0x9b,0x99,0x9f,0x9d,0x93,0x91,0x97,0x95,0x8b,0x89,0x8f,0x8d,0x83,0x81,0x87,0x85,
   0xbb,0xb9,0xbf,0xbd,0xb3,0xb1,0xb7,0xb5,0xab,0xa9,0xaf,0xad,0xa3,0xa1,0xa7,0xa5,
   0xdb,0xd9,0xdf,0xdd,0xd3,0xd1,0xd7,0xd5,0xcb,0xc9,0xcf,0xcd,0xc3,0xc1,0xc7,0xc5,
   0xfb,0xf9,0xff,0xfd,0xf3,0xf1,0xf7,0xf5,0xeb,0xe9,0xef,0xed,0xe3,0xe1,0xe7,0xe5
};

unsigned char mul_3[] =
{
   0x00,0x03,0x06,0x05,0x0c,0x0f,0x0a,0x09,0x18,0x1b,0x1e,0x1d,0x14,0x17,0x12,0x11,
   0x30,0x33,0x36,0x35,0x3c,0x3f,0x3a,0x39,0x28,0x2b,0x2e,0x2d,0x24,0x27,0x22,0x21,
   0x60,0x63,0x66,0x65,0x6c,0x6f,0x6a,0x69,0x78,0x7b,0x7e,0x7d,0x74,0x77,0x72,0x71,
   0x50,0x53,0x56,0x55,0x5c,0x5f,0x5a,0x59,0x48,0x4b,0x4e,0x4d,0x44,0x47,0x42,0x41,
   0xc0,0xc3,0xc6,0xc5,0xcc,0xcf,0xca,0xc9,0xd8,0xdb,0xde,0xdd,0xd4,0xd7,0xd2,0xd1,
   0xf0,0xf3,0xf6,0xf5,0xfc,0xff,0xfa,0xf9,0xe8,0xeb,0xee,0xed,0xe4,0xe7,0xe2,0xe1,
   0xa0,0xa3,0xa6,0xa5,0xac,0xaf,0xaa,0xa9,0xb8,0xbb,0xbe,0xbd,0xb4,0xb7,0xb2,0xb1,
   0x90,0x93,0x96,0x95,0x9c,0x9f,0x9a,0x99,0x88,0x8b,0x8e,0x8d,0x84,0x87,0x82,0x81,
   0x9b,0x98,0x9d,0x9e,0x97,0x94,0x91,0x92,0x83,0x80,0x85,0x86,0x8f,0x8c,0x89,0x8a,
   0xab,0xa8,0xad,0xae,0xa7,0xa4,0xa1,0xa2,0xb3,0xb0,0xb5,0xb6,0xbf,0xbc,0xb9,0xba,
   0xfb,0xf8,0xfd,0xfe,0xf7,0xf4,0xf1,0xf2,0xe3,0xe0,0xe5,0xe6,0xef,0xec,0xe9,0xea,
   0xcb,0xc8,0xcd,0xce,0xc7,0xc4,0xc1,0xc2,0xd3,0xd0,0xd5,0xd6,0xdf,0xdc,0xd9,0xda,
   0x5b,0x58,0x5d,0x5e,0x57,0x54,0x51,0x52,0x43,0x40,0x45,0x46,0x4f,0x4c,0x49,0x4a,
   0x6b,0x68,0x6d,0x6e,0x67,0x64,0x61,0x62,0x73,0x70,0x75,0x76,0x7f,0x7c,0x79,0x7a,
   0x3b,0x38,0x3d,0x3e,0x37,0x34,0x31,0x32,0x23,0x20,0x25,0x26,0x2f,0x2c,0x29,0x2a,
   0x0b,0x08,0x0d,0x0e,0x07,0x04,0x01,0x02,0x13,0x10,0x15,0x16,0x1f,0x1c,0x19,0x1a
};

unsigned char mul_9[] =
{
   0x00,0x09,0x12,0x1b,0x24,0x2d,0x36,0x3f,0x48,0x41,0x5a,0x53,0x6c,0x65,0x7e,0x77,
   0x90,0x99,0x82,0x8b,0xb4,0xbd,0xa6,0xaf,0xd8,0xd1,0xca,0xc3,0xfc,0xf5,0xee,0xe7,
   0x3b,0x32,0x29,0x20,0x1f,0x16,0x0d,0x04,0x73,0x7a,0x61,0x68,0x57,0x5e,0x45,0x4c,
   0xab,0xa2,0xb9,0xb0,0x8f,0x86,0x9d,0x94,0xe3,0xea,0xf1,0xf8,0xc7,0xce,0xd5,0xdc,
   0x76,0x7f,0x64,0x6d,0x52,0x5b,0x40,0x49,0x3e,0x37,0x2c,0x25,0x1a,0x13,0x08,0x01,
   0xe6,0xef,0xf4,0xfd,0xc2,0xcb,0xd0,0xd9,0xae,0xa7,0xbc,0xb5,0x8a,0x83,0x98,0x91,
   0x4d,0x44,0x5f,0x56,0x69,0x60,0x7b,0x72,0x05,0x0c,0x17,0x1e,0x21,0x28,0x33,0x3a,
   0xdd,0xd4,0xcf,0xc6,0xf9,0xf0,0xeb,0xe2,0x95,0x9c,0x87,0x8e,0xb1,0xb8,0xa3,0xaa,
   0xec,0xe5,0xfe,0xf7,0xc8,0xc1,0xda,0xd3,0xa4,0xad,0xb6,0xbf,0x80,0x89,0x92,0x9b,
   0x7c,0x75,0x6e,0x67,0x58,0x51,0x4a,0x43,0x34,0x3d,0x26,0x2f,0x10,0x19,0x02,0x0b,
   0xd7,0xde,0xc5,0xcc,0xf3,0xfa,0xe1,0xe8,0x9f,0x96,0x8d,0x84,0xbb,0xb2,0xa9,0xa0,
   0x47,0x4e,0x55,0x5c,0x63,0x6a,0x71,0x78,0x0f,0x06,0x1d,0x14,0x2b,0x22,0x39,0x30,
   0x9a,0x93,0x88,0x81,0xbe,0xb7,0xac,0xa5,0xd2,0xdb,0xc0,0xc9,0xf6,0xff,0xe4,0xed,
   0x0a,0x03,0x18,0x11,0x2e,0x27,0x3c,0x35,0x42,0x4b,0x50,0x59,0x66,0x6f,0x74,0x7d,
   0xa1,0xa8,0xb3,0xba,0x85,0x8c,0x97,0x9e,0xe9,0xe0,0xfb,0xf2,0xcd,0xc4,0xdf,0xd6,
   0x31,0x38,0x23,0x2a,0x15,0x1c,0x07,0x0e,0x79,0x70,0x6b,0x62,0x5d,0x54,0x4f,0x46
};

unsigned char mul_11[] =
{
   0x00,0x0b,0x16,0x1d,0x2c,0x27,0x3a,0x31,0x58,0x53,0x4e,0x45,0x74,0x7f,0x62,0x69,
   0xb0,0xbb,0xa6,0xad,0x9c,0x97,0x8a,0x81,0xe8,0xe3,0xfe,0xf5,0xc4,0xcf,0xd2,0xd9,
   0x7b,0x70,0x6d,0x66,0x57,0x5c,0x41,0x4a,0x23,0x28,0x35,0x3e,0x0f,0x04,0x19,0x12,
   0xcb,0xc0,0xdd,0xd6,0xe7,0xec,0xf1,0xfa,0x93,0x98,0x85,0x8e,0xbf,0xb4,0xa9,0xa2,
   0xf6,0xfd,0xe0,0xeb,0xda,0xd1,0xcc,0xc7,0xae,0xa5,0xb8,0xb3,0x82,0x89,0x94,0x9f,
   0x46,0x4d,0x50,0x5b,0x6a,0x61,0x7c,0x77,0x1e,0x15,0x08,0x03,0x32,0x39,0x24,0x2f,
   0x8d,0x86,0x9b,0x90,0xa1,0xaa,0xb7,0xbc,0xd5,0xde,0xc3,0xc8,0xf9,0xf2,0xef,0xe4,
   0x3d,0x36,0x2b,0x20,0x11,0x1a,0x07,0x0c,0x65,0x6e,0x73,0x78,0x49,0x42,0x5f,0x54,
   0xf7,0xfc,0xe1,0xea,0xdb,0xd0,0xcd,0xc6,0xaf,0xa4,0xb9,0xb2,0x83,0x88,0x95,0x9e,
   0x47,0x4c,0x51,0x5a,0x6b,0x60,0x7d,0x76,0x1f,0x14,0x09,0x02,0x33,0x38,0x25,0x2e,
   0x8c,0x87,0x9a,0x91,0xa0,0xab,0xb6,0xbd,0xd4,0xdf,0xc2,0xc9,0xf8,0xf3,0xee,0xe5,
   0x3c,0x37,0x2a,0x21,0x10,0x1b,0x06,0x0d,0x64,0x6f,0x72,0x79,0x48,0x43,0x5e,0x55,
   0x01,0x0a,0x17,0x1c,0x2d,0x26,0x3b,0x30,0x59,0x52,0x4f,0x44,0x75,0x7e,0x63,0x68,
   0xb1,0xba,0xa7,0xac,0x9d,0x96,0x8b,0x80,0xe9,0xe2,0xff,0xf4,0xc5,0xce,0xd3,0xd8,
   0x7a,0x71,0x6c,0x67,0x56,0x5d,0x40,0x4b,0x22,0x29,0x34,0x3f,0x0e,0x05,0x18,0x13,
   0xca,0xc1,0xdc,0xd7,0xe6,0xed,0xf0,0xfb,0x92,0x99,0x84,0x8f,0xbe,0xb5,0xa8,0xa3
};

unsigned char mul_13[] =
{
   0x00,0x0d,0x1a,0x17,0x34,0x39,0x2e,0x23,0x68,0x65,0x72,0x7f,0x5c,0x51,0x46,0x4b,
   0xd0,0xdd,0xca,0xc7,0xe4,0xe9,0xfe,0xf3,0xb8,0xb5,0xa2,0xaf,0x8c,0x81,0x96,0x9b,
   0xbb,0xb6,0xa1,0xac,0x8f,0x82,0x95,0x98,0xd3,0xde,0xc9,0xc4,0xe7,0xea,0xfd,0xf0,
   0x6b,0x66,0x71,0x7c,0x5f,0x52,0x45,0x48,0x03,0x0e,0x19,0x14,0x37,0x3a,0x2d,0x20,
   0x6d,0x60,0x77,0x7a,0x59,0x54,0x43,0x4e,0x05,0x08,0x1f,0x12,0x31,0x3c,0x2b,0x26,
   0xbd,0xb0,0xa7,0xaa,0x89,0x84,0x93,0x9e,0xd5,0xd8,0xcf,0xc2,0xe1,0xec,0xfb,0xf6,
   0xd6,0xdb,0xcc,0xc1,0xe2,0xef,0xf8,0xf5,0xbe,0xb3,0xa4,0xa9,0x8a,0x87,0x90,0x9d,
   0x06,0x0b,0x1c,0x11,0x32,0x3f,0x28,0x25,0x6e,0x63,0x74,0x79,0x5a,0x57,0x40,0x4d,
   0xda,0xd7,0xc0,0xcd,0xee,0xe3,0xf4,0xf9,0xb2,0xbf,0xa8,0xa5,0x86,0x8b,0x9c,0x91,
   0x0a,0x07,0x10,0x1d,0x3e,0x33,0x24,0x29,0x62,0x6f,0x78,0x75,0x56,0x5b,0x4c,0x41,
   0x61,0x6c,0x7b,0x76,0x55,0x58,0x4f,0x42,0x09,0x04,0x13,0x1e,0x3d,0x30,0x27,0x2a,
   0xb1,0xbc,0xab,0xa6,0x85,0x88,0x9f,0x92,0xd9,0xd4,0xc3,0xce,0xed,0xe0,0xf7,0xfa,
   0xb7,0xba,0xad,0xa0,0x83,0x8e,0x99,0x94,0xdf,0xd2,0xc5,0xc8,0xeb,0xe6,0xf1,0xfc,
   0x67,0x6a,0x7d,0x70,0x53,0x5e,0x49,0x44,0x0f,0x02,0x15,0x18,0x3b,0x36,0x21,0x2c,
   0x0c,0x01,0x16,0x1b,0x38,0x35,0x22,0x2f,0x64,0x69,0x7e,0x73,0x50,0x5d,0x4a,0x47,
   0xdc,0xd1,0xc6,0xcb,0xe8,0xe5,0xf2,0xff,0xb4,0xb9,0xae,0xa3,0x80,0x8d,0x9a,0x97
};

unsigned char mul_14[] =
{
   0x00,0x0e,0x1c,0x12,0x38,0x36,0x24,0x2a,0x70,0x7e,0x6c,0x62,0x48,0x46,0x54,0x5a,
   0xe0,0xee,0xfc,0xf2,0xd8,0xd6,0xc4,0xca,0x90,0x9e,0x8c,0x82,0xa8,0xa6,0xb4,0xba,
   0xdb,0xd5,0xc7,0xc9,0xe3,0xed,0xff,0xf1,0xab,0xa5,0xb7,0xb9,0x93,0x9d,0x8f,0x81,
   0x3b,0x35,0x27,0x29,0x03,0x0d,0x1f,0x11,0x4b,0x45,0x57,0x59,0x73,0x7d,0x6f,0x61,
   0xad,0xa3,0xb1,0xbf,0x95,0x9b,0x89,0x87,0xdd,0xd3,0xc1,0xcf,0xe5,0xeb,0xf9,0xf7,
   0x4d,0x43,0x51,0x5f,0x75,0x7b,0x69,0x67,0x3d,0x33,0x21,0x2f,0x05,0x0b,0x19,0x17,
   0x76,0x78,0x6a,0x64,0x4e,0x40,0x52,0x5c,0x06,0x08,0x1a,0x14,0x3e,0x30,0x22,0x2c,
   0x96,0x98,0x8a,0x84,0xae,0xa0,0xb2,0xbc,0xe6,0xe8,0xfa,0xf4,0xde,0xd0,0xc2,0xcc,
   0x41,0x4f,0x5d,0x53,0x79,0x77,0x65,0x6b,0x31,0x3f,0x2d,0x23,0x09,0x07,0x15,0x1b,
   0xa1,0xaf,0xbd,0xb3,0x99,0x97,0x85,0x8b,0xd1,0xdf,0xcd,0xc3,0xe9,0xe7,0xf5,0xfb,
   0x9a,0x94,0x86,0x88,0xa2,0xac,0xbe,0xb0,0xea,0xe4,0xf6,0xf8,0xd2,0xdc,0xce,0xc0,
   0x7a,0x74,0x66,0x68,0x42,0x4c,0x5e,0x50,0x0a,0x04,0x16,0x18,0x32,0x3c,0x2e,0x20,
   0xec,0xe2,0xf0,0xfe,0xd4,0xda,0xc8,0xc6,0x9c,0x92,0x80,0x8e,0xa4,0xaa,0xb8,0xb6,
   0x0c,0x02,0x10,0x1e,0x34,0x3a,0x28,0x26,0x7c,0x72,0x60,0x6e,0x44,0x4a,0x58,0x56,
   0x37,0x39,0x2b,0x25,0x0f,0x01,0x13,0x1d,0x47,0x49,0x5b,0x55,0x7f,0x71,0x63,0x6d,
   0xd7,0xd9,0xcb,0xc5,0xef,0xe1,0xf3,0xfd,0xa7,0xa9,0xbb,0xb5,0x9f,0x91,0x83,0x8d
};

unsigned char rcon[11] =
{
   0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36,
};

unsigned char * g (unsigned char wInput[4], int counter)
{
   unsigned char * wReady = malloc(4);
   unsigned char temp[4] = "";
   unsigned char a = wInput[0];
   for(int i =0;i<3; i++)
   {
       temp[i] = wInput[(i+1)];
   }
   temp[3] = a;

   for (int i =0; i<4;i++)
       temp[i] = s[temp[i]];

   //unsigned char array formed for xoring with rcon

   unsigned char array2[4] = "";
   array2[0] = rcon[counter];
   array2[1] = array2[2] = array2[3] = 0x00;

   for (int i=0;i<4;i++)
   wReady[i] = temp[i] ^ array2[i];
   return wReady;
}

unsigned char * keyExpansion(unsigned char key[16])
{
   unsigned char words[44][4];
   for (int i = 0; i < 44; ++i)
   {
       for (int j = 0; j <4; ++j)
       {
           words[i][j]=0x00;
       }
   }

   unsigned char * expandedKey = malloc(176);

   int byteCount = 0; //this is to keep a count on the bytes of the expandedKey array

   for (int i=0;i<16;i++)
           expandedKey[i] = key[i];

   for(int j=0;j<4;j++)
   {
        for(int k=0;k<4;k++)
        {
           words[j][k] = expandedKey[byteCount];
           byteCount++;
        }
   }
   for(int l=4;l<44;l++)
   {
       if((l%4)==0)
       {
           for(int m=0;m<4;m++)
           {
               words[l][m] = words[(l-4)][m] ^ g(words[l-1], (l/4))[m];
           }
       }
       else
       {
           for(int n=0;n<4;n++)
           {
               words[l][n] = words[l-1][n] ^ words[l-4][n];
           }
       }
   }

   int loc=0;
   for(int i=0;i<44;i++ )
   {
       for(int j=0;j<4;j++)
       {
           expandedKey[loc] = words[i][j];
           loc++;
       }
   }
   return expandedKey;
}

void mixColumns(unsigned char * plainText)
{
   unsigned char * tempC = malloc(16);

   for (int i = 0; i < 4; ++i)
   {
       tempC[(4*i)+0] = (unsigned char) (mul2[plainText[(4*i)+0]] ^ mul_3[plainText[(4*i)+1]] ^ plainText[(4*i)+2] ^
plainText[(4*i)+3]);
       tempC[(4*i)+1] = (unsigned char) (plainText[(4*i)+0] ^ mul2[plainText[(4*i)+1]] ^ mul_3[plainText[(4*i)+2]] ^
plainText[(4*i)+3]);
       tempC[(4*i)+2] = (unsigned char) (plainText[(4*i)+0] ^ plainText[(4*i)+1] ^ mul2[plainText[(4*i)+2]] ^
mul_3[plainText[(4*i)+3]]);
       tempC[(4*i)+3] = (unsigned char) (mul_3[plainText[(4*i)+0]] ^ plainText[(4*i)+1] ^ plainText[(4*i)+2] ^
mul2[plainText[(4*i)+3]]);
   }

   for (int i = 0; i < 16; ++i)
   {
       plainText[i] = tempC[i];
   }
   free(tempC);
}
void inverseMixedColumn (unsigned char * plainText)
{
   unsigned char * tempC = malloc(18);

   for (int i = 0; i < 4; ++i)
   {
       tempC[(4*i)+0] = (unsigned char) (mul_14[plainText[(4*i)+0]] ^ mul_11[plainText[(4*i)+1]] ^
mul_13[plainText[(4*i)+2]] ^ mul_9[plainText[(4*i)+3]]);
```

```
        tempC[(4*i)+1] = (unsigned char) (mul_9[plainText[(4*i)+0]] ^ mul_14[plainText[(4*i)+1]] ^
mul_11[plainText[(4*i)+2]] ^ mul_13[plainText[(4*i)+3]]);
        tempC[(4*i)+2] = (unsigned char) (mul_13[plainText[(4*i)+0]] ^ mul_9[plainText[(4*i)+1]] ^
mul_14[plainText[(4*i)+2]] ^ mul_11[plainText[(4*i)+3]]);
        tempC[(4*i)+3] = (unsigned char) (mul_11[plainText[(4*i)+0]] ^ mul_13[plainText[(4*i)+1]] ^
mul_9[plainText[(4*i)+2]] ^ mul_14[plainText[(4*i)+3]]);
    }
    for (int i = 0; i < 16; ++i)
    {
        plainText[i] = tempC[i];
    }
    free(tempC);
}
void byteSubShiftRow(unsigned char * state)
{

    unsigned char tmp[16];

    tmp[0] = s[state[0]];
    tmp[1] = s[state[5]];
    tmp[2] = s[state[10]];
    tmp[3] = s[state[15]];

    tmp[4] = s[state[4]];
    tmp[5] = s[state[9]];
    tmp[6] = s[state[14]];
    tmp[7] = s[state[3]];

    tmp[8] = s[state[8]];
    tmp[9] = s[state[13]];
    tmp[10] = s[state[2]];
    tmp[11] = s[state[7]];

    tmp[12] = s[state[12]];
    tmp[13] = s[state[1]];
    tmp[14] = s[state[6]];
    tmp[15] = s[state[11]];

    for(int i=0;i<16;i++)
    {
        state[i] = tmp[i];
    }
}
void inverseByteSubShiftRow(unsigned char * plainText)
{
    unsigned char * temp = malloc(16);
    temp[0] = inv_s[plainText[0]];
    temp[1] = inv_s[plainText[13]];
    temp[2] = inv_s[plainText[10]];
    temp[3] = inv_s[plainText[7]];
    temp[4] = inv_s[plainText[4]];
    temp[5] = inv_s[plainText[1]];
    temp[6] = inv_s[plainText[14]];
    temp[7] = inv_s[plainText[11]];
    temp[8] = inv_s[plainText[8]];
    temp[9] = inv_s[plainText[5]];
    temp[10] = inv_s[plainText[2]];
    temp[11] = inv_s[plainText[15]];
    temp[12] = inv_s[plainText[12]];
    temp[13] = inv_s[plainText[9]];
    temp[14] = inv_s[plainText[6]];
    temp[15] = inv_s[plainText[3]];

    for (int i = 0; i < 16; ++i)
        plainText[i] = temp[i];

    free(temp);
}

unsigned char * AESDecryption(unsigned char * cipher, unsigned char * expandedKey)
{
    unsigned char * state = malloc(16);
    unsigned char * plainTxt = malloc(16);
    //key whitening
    for (int i = 0; i < 16; ++i)
        state[i] = cipher[i] ^ expandedKey[160+i];

    // 9 rounds of decryption
    for (int rounds = 9; rounds >0 ; rounds--)
    {
        inverseByteSubShiftRow(state);
        int counter = 0;
        int loc = 16*rounds;
        while(counter<16)
        {
            state[counter] ^= expandedKey[loc];
            loc++;
            counter++;
        }
        inverseMixedColumn(state);
    }

    //final 10th round of decryption
    inverseByteSubShiftRow(state);
    for(int i =0; i<16; i++)
    {
        plainTxt[i] = state[i] ^ expandedKey[i];
        //printf("\n plainText[%d]=%x",i,plainText[i]);
    }
    free(state);
    return plainTxt;
}

int main(int argc, char *argv[])
{
            char sender[INET6_ADDRSTRLEN];
            int sockfd, ret, i;
            int sockopt,c_index=0,h_index=0,flag=0,dig=0,p=0,j=0,q=0,count=1;
            ssize_t numbytes;
            struct ifreq ifopts; /* set promiscuous mode */
            struct ifreq if_ip;   /* get ip addr */
            struct sockaddr_storage their_addr;
        unsigned char * expandedKey = malloc(176);
        unsigned char * cipher1 = malloc(16);
        unsigned char * plainText1 = malloc(16);
        unsigned char * plainText= malloc(1024);
            unsigned char * hash_recv= malloc(32);
        unsigned char * hash;
        unsigned char * hash_string= malloc(32);
        unsigned char * buf = malloc(1024);
        unsigned char * cipher = malloc(144);
        unsigned char * cipher_str = malloc(288);
        //unsigned char hash_string[32];
            //unsigned char buf[BUF_SIZ],cipher[144],cipher_str[288];
        //uint8_t = unsigned char data type
            char ifName[IFNAMSIZ];
        unsigned char key[14]= { 0x32, 0x21, 0x23, 0x52, 0x71, 0x98, 0x24, 0x03, 0x38, 0x27, 0x01, 0x12, 0x95,
0x23};
        double begin,end,time_spent,time_hmac=0,time_comp=0,time_decrypt=0,t=0;

        expandedKey = keyExpansion(key);

            /* Get interface name */
            if (argc > 1)
                    strcpy(ifName, argv[1]);
            else
                    strcpy(ifName, DEFAULT_IF);

            /* Header structures */
            struct ether_header *eh = (struct ether_header *) buf;
            struct iphdr *iph = (struct iphdr *) (buf + sizeof(struct ether_header));
            struct udphdr *udph = (struct udphdr *) (buf + sizeof(struct iphdr) + sizeof(struct ether_header));

            memset(&if_ip, 0, sizeof(struct ifreq));

            /* Open PF_PACKET socket, listening for EtherType ETHER_TYPE */
            if ((sockfd = socket(PF_PACKET, SOCK_RAW, htons(ETHER_TYPE))) == -1) {
                    perror("listener: socket");
                    return -1;
            }

            /* Set interface to promiscuous mode - do we need to do this every time? */
            strncpy(ifopts.ifr_name, ifName, IFNAMSIZ-1);
            ioctl(sockfd, SIOCGIFFLAGS, &ifopts);
            ifopts.ifr_flags |= IFF_PROMISC;
            ioctl(sockfd, SIOCSIFFLAGS, &ifopts);
            /* Allow the socket to be reused - incase connection is closed prematurely */
            if (setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &sockopt, sizeof sockopt) == -1) {
                    perror("setsockopt");
                    close(sockfd);
                    exit(EXIT_FAILURE);
            }
            /* Bind to device */
            if (setsockopt(sockfd, SOL_SOCKET, SO_BINDTODEVICE, ifName, IFNAMSIZ-1) == -1) {
                    perror("SO_BINDTODEVICE");
                    close(sockfd);
                    exit(EXIT_FAILURE);
            }

repeat: printf("listener: Waiting to recvfrom...\n");
            numbytes = recvfrom(sockfd, buf, BUF_SIZ, 0, NULL, NULL);
            printf("listener: got packet %lu bytes\n", numbytes);

            /* Check the packet is for me */
            if (eh->ether_dhost[0] == DEST_MAC0 &&
                            eh->ether_dhost[1] == DEST_MAC1 &&
                            eh->ether_dhost[2] == DEST_MAC2 &&
                            eh->ether_dhost[3] == DEST_MAC3 &&
                            eh->ether_dhost[4] == DEST_MAC4 &&
                            eh->ether_dhost[5] == DEST_MAC5) {
```

```
                printf("Correct destination MAC address\n");
        } else {
                printf("Wrong destination MAC: %x:%x:%x:%x:%x:%x\n",
                                                        eh->ether_dhost[0],
                                                        eh->ether_dhost[1],
                                                        eh->ether_dhost[2],
                                                        eh->ether_dhost[3],
                                                        eh->ether_dhost[4],
                                                        eh->ether_dhost[5]);
                ret = -1;
                goto done;
        }

        /* Get source IP */
        ((struct sockaddr_in *)&their_addr)->sin_addr.s_addr = iph->saddr;
        inet_ntop(AF_INET, &((struct sockaddr_in*)&their_addr)->sin_addr, sender, sizeof sender);

        /* Look up my device IP addr if possible */
        strncpy(if_ip.ifr_name, ifName, IFNAMSIZ-1);
        if (ioctl(sockfd, SIOCGIFADDR, &if_ip) >= 0) { /* if we can't check then don't */
                printf("Source IP: %s\n My IP: %s\n", sender,
                                inet_ntoa(((struct sockaddr_in *)&if_ip.ifr_addr)->sin_addr));
                /* ignore if I sent it */
                if (strcmp(sender, inet_ntoa(((struct sockaddr_in *)&if_ip.ifr_addr)->sin_addr)) == 0){
                        printf("but I sent it :(\n");
                        ret = -1;
                        goto done;
                }
        }

        /* UDP payload length */
        ret = ntohs(udph->len) - sizeof(struct udphdr);

        /* Print packet */
        printf("\n \t Received Packet Data:\n");
    for (i=0; i<numbytes; i++)
    {
      printf("%02x:", buf[i]);
     }
    printf("\n\n received cipher text:\n");
    for(i=22;i<=165;i++)
    {
       printf(" %02x",buf[i]);
       cipher[c_index++]=buf[i];
    }
```

```
        c_index=0;
        printf("\n\n Received hash value:\n");
        for(i=numbytes-32;i<numbytes;i++)
        {
            printf(" %02x",buf[i]);
            hash_recv[h_index++]=buf[i];
        }
        h_index=0;
        printf("\n\n Received cipher text in a variable extracted from packet data:\n");
        for(i=0;i<144;i++)
        {
            printf(" %02x",cipher[c_index++]);
        }
        c_index=0;
        printf("\n\n Received hash value in a variable extracted packet data:\n");
        h_index=0;
        for(i=0;i<32;i++)
        {
            printf(" %02x",hash_recv[h_index++]);
        }
        h_index=0;
        for (i=0;i<144;i++)
             {
                        sprintf( &(cipher_str [i*2]), "%02x", cipher[i]);
        }
        //printf(" Cipher Text string :\n %s", cipher_str);
                        begin = clock();
                        hash = HMAC(EVP_sha256(), key, strlen((char *)key), cipher_str, strlen((char *)cipher_str),
NULL, NULL);
                        end = clock();
                        time_hmac= (double)(end - begin) / CLOCKS_PER_SEC;

                        begin = clock();
                        hash = HMAC(EVP_sha256(), key, strlen((char *)key), cipher_str, strlen((char *)cipher_str),
NULL, NULL);
                        end = clock();
                        time_hmac= (double)(end - begin) / CLOCKS_PER_SEC;

              //printf(" \n time-spent hmac  =%lf",time_hmac*1000);
                        for (i = 0; i < 32 ; i++)
                                    sprintf(&(hash_string[i * 2]), "%02x", hash[i]);
              printf("\n\n Generated hash string at the receiver:%s\n",hash_string);

                        begin=clock();
                        for (i = 0; i < 32 ; i++)
```

```
{
                        //printf(" hash value: \n %02x",hash_recv[h_index]);
                        //printf("  %02x\n",hash[i]);

            if( hash_recv[h_index++] != hash[i])
                                    flag=1;
}
            end=clock();
            time_comp= (double)(end - begin) / CLOCKS_PER_SEC;
      //printf(" \n time-spent comparision  =%lf",time_comp*1000);

            printf("\n flag value=%d",flag);
            dig=0;
            if (flag == 0)
{
            printf("\n ** Hash values matched **\n");
                        for ( i=0; i<9; i++)
                        {
                                    for (q=0; q<16; q++)
                                    cipher1[q]= cipher[j++];
                        begin = clock();
                        plainText1=AESDecryption(cipher1,expandedKey);
                                    end = clock();
                                    time_spent= (double)(end - begin) / CLOCKS_PER_SEC;
                        time_decrypt = time_decrypt+time_spent;
                        for (q=0; q<16; q++)
                                    {
                                    plainText[p]= plainText1[q];
                                                //printf(" %.2x", plainText[p++]);
                                    }
                                    //printf("\n");
            }
                        //printf("\n Decryption time =%lf\n",time_decrypt*1000);

}
else
            {
            //printf("\n Hash values mis-matched\n");

            }



            j=0;p=0;
            strcpy(cipher1,"\0");
            strcpy(plainText1,"\0");
```

```
                strcpy(plainText,"\0");
        //printf("packet=%d",count++);
        printf(" \n MAC Generation time using HMAC-SHA256=%lf",time_hmac*1000);
        printf(" \n MAC comparision time=%lf",time_comp*1000);
        printf(" \n Decryption time=%lf\n",time_decrypt*1000);
        printf("\n-------------------------------------\n\n");
        //printf(" Total time  =%lf", (time_hmac+time_comp+time_decrypt) );
        time_hmac=0;time_comp=0;time_decrypt=0;
done:   goto repeat;

        close(sockfd);
        return ret;
}
```

APPENDIX G

Attached is the Wireshark capture of the Sampled Value packets from the sending device.

No.	Time	Time delta	Source	Destination	Protocol	Length
1	0	0	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
2	1,000730208	1,000730208	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
3	1,000567512	1,000567512	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
4	1,019855342	1,019855342	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
8	1,001075039	0,040264758	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
10	1,000648991	0,506195143	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
11	1,000570027	1,000570027	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
16	1,000876031	0,152832343	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
21	1,000706747	0,114443281	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
26	1,001868698	0,075977039	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176

27	1,000334268	1,000334268	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
28	1,00058097	1,00058097	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
29	1,017931034	1,017931034	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
31	1,000717172	0,524779632	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
33	1,000927511	0,908430205	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
36	1,027336791	0,046070561	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
40	1,006099577	0,058297948	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
41	1,000746823	1,000746823	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
42	1,001267288	1,001267288	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
45	1,000555793	0,051297535	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
46	1,014830343	1,014830343	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
48	1,001134239	0,573440132	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176

49	1,001399845	1,001399845	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
50	1,001134302	1,001134302	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
51	1,000769618	1,000769618	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
52	1,001289992	1,001289992	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
53	1,000800083	1,000800083	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
54	1,011678912	1,011678912	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
55	1,000883413	1,000883413	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
57	1,001945973	0,58991945	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
58	1,002937633	1,002937633	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
59	1,001064942	1,001064942	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
60	1,001182593	1,001182593	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
61	1,000573951	1,000573951	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176

62	1,003486131	1,003486131	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
63	1,002829743	1,002829743	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
65	1,023445645	0,039671099	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
67	1,001097158	0,623873434	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
68	1,009539806	1,009539806	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
70	1,000454155	0,049614454	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
71	1,004300996	1,004300996	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
72	1,001235977	1,001235977	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
74	1,001089241	0,055718282	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
76	1,001400657	0,300235173	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
77	1,004629204	1,004629204	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176

APPENDIX H

Attached is the Wireshark capture of the Sampled Value packets from the receiving device.

No.	Time	Source	Destination	Protocol	Length
1	0.000000000	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
5	1.001302427	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
7	1.000630852	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
8	1.000355431	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
13	1.000876148	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
18	1.000772328	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
23	1.001844746	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
24	1.000285075	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
25	1.000931510	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
26	1.017946027	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176

28	1.000394877	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
30	1.001198878	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
33	1.027394484	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
37	1.005958011	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
38	1.000609014	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
39	1.001205644	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
42	1.000941860	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
43	1.014430880	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
45	1.001542392	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
46	1.001165932	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
47	1.001046078	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
48	1.000869278	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176

49	1.001141924	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
50	1.000891396	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
51	1.012036862	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
52	1.000375548	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
54	1.002338513	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
55	1.002676293	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
56	1.001021821	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
57	1.001131982	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
58	1.000966325	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
59	1.002994688	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
60	1.003122324	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
62	1.023360673	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176

64	1.001145123	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
65	1.009552273	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
67	1.000342314	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
68	1.004276707	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
69	1.001357436	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
71	1.000897365	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
73	1.001407896	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
74	1.004875002	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176
77	1.000592307	HewlettP_c5:77:a1	PcsCompu_f8:42:a7	IEC61850 Sampled Values	176

APPENDIX I

Attached are the computational times of generating MAC and encryption for Sampled Value packets from the sending device.

No.	MAC Generation Time (ms)	Encryption Time (ms)
1	0,002	0,008
2	0,002	0,008
3	0,002	0,01
4	0,002	0,008
5	0,002	0,008
6	0,002	0,01
7	0,002	0,008
8	0,002	0,009
9	0,002	0,01
10	0,002	0,009
11	0,002	0,008
12	0,002	0,009
13	0,002	0,01
14	0,002	0,011
15	0,002	0,008
16	0,003	0,014
17	0,004	0,014
18	0,002	0,011
19	0,002	0,01
20	0,002	0,01
21	0,002	0,01
22	0,002	0,01
23	0,002	0,011
24	0,002	0,008
25	0,002	0,01
26	0,002	0,009
27	0,002	0,018
28	0,002	0,01
29	0,002	0,011

30	0,003	0,01
31	0,004	0,015
32	0,003	0,011
33	0,002	0,01
34	0,003	0,012
35	0,004	0,014
36	0,002	0,01
37	0,002	0,009
38	0,002	0,009
39	0,002	0,009
40	0,002	0,01
41	0,002	0,008
42	0,002	0,009
43	0,002	0,008
44	0,002	0,021
45	0,002	0,019
46	0,003	0,008
47	0,002	0,009
48	0,003	0,008
49	0,002	0,008
50	0,003	0,015
	Average (MAC Generation)	Average (Encryption)
	0,00226	0,01044

APPENDIX J

Attached are the computational times of generating MAC and decryption for Sampled Value packets from the receiving device.

No.	MAC Generation Time (ms)	MAC comparison time (ms)	Decryption time
1	0,002	0,000	0,001
2	0,005	0,001	0,001
3	0,012	0,001	0,001
4	0,011	0,002	0,001
5	0,011	0,002	0,001
6	0,011	0,002	0,001
7	0,008	0,001	0,001
8	0,007	0,001	0,001
9	0,011	0,002	0,001
10	0,019	0,002	0,001
11	0,011	0,001	0,001
12	0,011	0,002	0,001
13	0,007	0,001	0,001
14	0,011	0,002	0,001
15	0,012	0,001	0,001
16	0,011	0,002	0,001
17	0,007	0,002	0,001
18	0,018	0,002	0,001
19	0,006	0,001	0,001
20	0,001	0,001	0,001
21	0,007	0,000	0,001
22	0,011	0,001	0,001
23	0,011	0,002	0,001
24	0,012	0,002	0,001
25	0,001	0,000	0,001
26	0,002	0,000	0,001
27	0,002	0,000	0,001

28	0,003	0,000	0,001
29	0,002	0,001	0,001
30	0,006	0,000	0,001
31	0,011	0,001	0,001
32	0,009	0,002	0,001
33	0,011	0,001	0,001
34	0,011	0,002	0,001
35	0,011	0,002	0,001
36	0,007	0,001	0,001
37	0,002	0,000	0,001
38	0,005	0,001	0,001
39	0,002	0,000	0,001
40	0,002	0,000	0,001
41	0,003	0,000	0,001
42	0,002	0,001	0,001
43	0,006	0,000	0,001
44	0,011	0,001	0,001
45	0,009	0,002	0,001
46	0,011	0,001	0,001
47	0,011	0,002	0,001
48	0,011	0,002	0,001
49	0,006	0,001	0,001
50	0,006	0,001	0,001
	Average (MAC Generation)	Average (MAC Comparison)	Average (Decryption)
	0,008	0,001	0,001