Brno University of Technology
Faculty of Information Technology

Security of Smart Grid Communication

Habilitation

Ing. Petr Matoušek, Ph.D., M.A.

Brno, 2021

Abstract

Protection of industrial communication systems against cyber attacks has become a great challenge during the past years due to the convergence of Operational Technologies (OT) and Information Technologies (IT), adoption of the TCP/IP to industrial networks, and the rising level of automation and intelligent control of industrial processes. Security and safety of critical infrastructure systems that include power plants, substations, water and gas distribution, traffic control systems, etc., can be implemented on various levels. In this work we focus on security of industrial system via high-level communication monitoring and automated anomaly detection. In this work, we deal specifically with smart grid communication protocols like IEC 104, GOOSE, and MMS.

At first, we address the issue of high-level visibility of industrial communication. By monitoring of transmitted commands and their parameters we can disclose real activities of the system and identify potential threats. For this task we adopt Netflow/IPFIX technology that is a standard for IP networks. The adoption requires re-definition of the flow and extension of IPFIX records by values obtained from industrial protocols, e.g., transmitted operations (setting on/off the switch breaker, reading process values), device status, types and identification of target objects, etc. Monitoring data are further analyzed and used for anomaly detection.

Industrial devices are usually pre-configured and their communication exhibits stable device-specific patterns. When we learn these patterns, we can identify unusual behavior. This work introduces two novel techniques for anomaly detection of smart grid communication. The first one is based on probabilistic automata that model typical communication sequences. Using this approach we can detect unexpected message sequences, unusual command frequency, or irregular data exchanges. Such anomalies may indicate specific cyber attacks or device failures.

The second presented technique observes time properties of packets. Using statistical methods we model typical distribution of packet inter-arrival times and create statistical profiles that represent normal behavior of the network. By applying the Three Sigma Rule we observe deviation from the learned profile.

By combination of both techniques we are able to detect common anomalies in the smart grid communication and improve security of smart grid networks. The presented methods were implemented and become a part of commercial solution.

Preface

My son, beware of anything beyond these.
Of making many books there is no end,
and much study is a weariness of the flesh.

The end of the matter; all has been heard.
Fear God and keep his commandments,
for this is the whole duty of man.

For God will bring every deed into judgment,
with every secret thing, whether good or evil.

Ecclesiastes

This work has been carried out at the Faculty of Information Technology, Brno University of Technology, Czech Republic during 2015-2021. I am expecially thankful to my colleague Ondřej Ryšavý for his advises and encouragement during this research, and also careful reading of the text and its corrections. Without his support I would not be able to finish it.

My great thanks belong to our research team NES@FIT for the mutual support, new ideas, challenges and feedback that are essential for any meaningful work. I also appreciate their friendship and help.

Finally, I would like to express my gratitude to my dear wife Olga for her constant support, patience and love.

Petr Matoušek
Brno, 16th September 2021

Contents

Chapter 1

Introduction

This work is focused on security of industrial communication in a domain of power systems and energy distribution with a specific application to smart grids. The smart grid (also called the power grid) produces and supplies electrical energy to customers over a large geographical region. It is an electricity network that includes energy generation using various resources (nuclear, water, wind, photovoltaic, etc.), energy transmission and distribution, advanced metering, and other operations. Automation and control of smart grids is implemented using Industrial Control System (ICS) and Supervisory Control and Data Acquisition (SCADA) communication protocols, e.g, IEC 60870-5-104 (aka IEC 104), DNP3, IEC 61850 MMS and GOOSE, or DLMS [72]. These protocols transmit monitoring data and control commands in substations, smart meters, or phasor measurement units. Proper operation of ICS/SCADA communication systems is essential for stability, reliability and security of smart grids.

Power production and energy distribution fall into the category of critical systems which interruption or blackouts may have fatal consequence on industrial processes, traffic, and even lives of individual persons. For this reason, we need to monitor smart grid communication and detect possible malfunctioning of a device, lost packets, communication delays, and also cyber attacks initiated from the outside of the smart grid network but also from the inside of the network from an infected machine [53].

Current security of smart grids focuses on the network edge where firewalls and IDS systems control and filter both outgoing and incoming traffic. This approach is successful against external threats, but have a little effect on internal threats initiated from infected machines. These threats are real and have significant consequences as seen in a cyber attack against Ukrainian power plants in 2016 [41] or a more recent ransomware attack against the Colonial Gas Pipeline in the U.S. that happened in May 2021 [106].

Security of smart grid communication includes two important steps: (i) real-time visibility of ICS communication via extended security monitoring, and (ii) automated detection of security incidents using advanced detection methods. In this work, we present a solution for both tasks. The extended visibility of smart grid communication can be implemented using flow-based monitoring of industrial protocols [108]. This technique using Netflow/IPFIX protocols [34] is commonly used for IT networks monitoring. For ICS communication, we need to re-define *the flow* and import specific features extracted from ICS packets to the network monitoring system so that it is able to reveal transmitted operations. This technique with special focus on smart grid communication protocols is described in Chap. 3.

The second step for enhancing cyber security of smart grid communication requires automated detection of security incidents. In this work we apply two approaches: *automata-based approach* and *statistical anomaly detection*. *Automata-based* approach uses language theory. It stems from the hypothesis that communication sequences between two industrial devices are stable and do not change over time. By observing device to device communication, we can learn their typical communication sequences and form a probabilistic language that represents a *communication profile* of these devices. For this task we employ two formalisms: *Prefix Trees* (PT) and *Deterministic Probabilistic Automata* (DPA). Both these formalisms represent an efficient way how to create a probabilistic language from a set of sample strings obtained from normal communication. This work demonstrates that communication profiles are effective for detection of common ICS security incidents like the command injection, packet manipulation, network scanning, or lost connection. An important advantage of this approach is that it uses the standardized IPFIX flow monitoring protocol and it can be easily incorporated into common security information and event management (SIEM) systems. Modeling ICS communication using probabilistic automata and anomaly detection based on automata models is described in Chapter 4.

Another approach to anomaly detection that we present in this work, observes timing properties of ICS communication. Since the ICS communication is stable, periodical and contains regular communication patterns, it can be described using statistical models. Here we deal with modeling selected features of ICS traffic, more specifically packet direction and inter-arrival times. Based on the previous research [97, 109] and our own experiments, we approximate inter-arrival times of ICS communication and the number of transmitted packets within a time window with normal distribution. Using selected features we create *a statistical profile* of an ICS communication. Our experiments show that using statistical profiles, we can detect various anomalies caused by irregular transmission, device or link failures, packet injection, scanning, or denial of service (DoS) attack.

Unlike automata-based approach, the statistical profiles do not require a deep analysis of ICS protocols during detection phase because they operate on packet level. Statistical profiles build a complementary view on smart grid communication security with respect to the automata-based model. We show how combination of these two presented approaches increases accuracy and flexibility of anomaly detection in smart grid networks. The statistical anomaly detection is described in Chapter 5.

1.1 Structure of the Text

The text of this work is structured as follows. Chapter 2 introduces typical industrial protocols deployed in smart grids, discusses their features and typical communication patterns of ICS communication. It also describes common cyber threats against ICS communication. It lists major cyber attacks against ICS systems that happened in previous years. How these attacks may operate is demonstrated on two cyber attacks against Ukrainian power grid in 2015 and 2016 where ICS communication was manipulated and misused by an attacker. To prevent such sophisticated attacks that are not visible by common security techniques is the main motivation of our research. Further in that chapter we discuss common ICS attack vectors defined by NIST Report 8219 [92] and MITRE ATT&CK matrix for Industrial Control Systems[1]. We conclude the chapter with mentioning main mitigation and prevention techniques that are commonly used in smart grid networks.

Chapter 3 deals with the first main task related to security of ICS communication, i.e., the increased visibility of ICS communication within the network. First, it reviews common ICS protocols deployed in smart grid networks and discusses their features and vulnerabilities. Further details of common ICS protocols are given in Appendix A. Our approach is built on IPFIX monitoring. For our purpose, we define *an ICS Flow* as a sequence of ICS packets with the same property that includes not only L3 and L4 meta data as Netflow/IPFIX records but also application data extracted from ICS protocol headers. By adding selected ICS header values to flow records, we increase visibility of ICS communication which is then used for anomaly detection. The chapter presents recommended ICS headers for common smart grid protocols IEC 104, MMS, GOOSE and DLMS. Then we show how extended ICS monitoring and analysis covers Behavior Anomaly Detection (BAD) capabilities defined by NIST Report 8219 [92].

Chapter 4 presents an automata-based technique for anomaly detection of ICS communication. Using extended ICS flow records, we obtain communication sequences of normal ICS traffic that is further used to create a probabilistic model of ICS communication using Prefix Trees (PT) and Deterministic Probabilistic Automata (DPA). While Prefix Trees are easy and fast to construct, DPAs provide more compact representation where similar states are merged using Alergia algorithm [39]. The result of learning is a probabilistic model in form of a PT or DPA that represents a typical communication between two ICS devices. Our results show that even for samples covering two-days communication, the resulting automaton is relatively small with tens of states at maximum. Having the probabilistic model of communication, we can detect anomalies by observing passing ICS sequences and comparing them with the learned models. For anomaly detection we implemented two approaches: single conversation reasoning where we compute probability for each conversation with respect to learned automata, and distribution reasoning where we create a DPA for each time window and com-

[1] See https://collaborate.mitre.org/attackics/index.php/Main_Page [07/2021]

pare this DPA with learned DPAs using 2-Euclidean distance. Our results prove that such approach is efficient to detect communication loss, rogue devices on the network, switching, scanning, and injection attacks. Probabilistic approach is not suitable to detect DoS attacks that use communication sequences present in the training dataset. However, a DoS attack can be easily detected by the statistical approach.

Chapter 5 presents a complementary approach to automated-based detection which uses statistical properties of ICS communication. Since ICS communication is stable, periodical and with regular communication patterns as observed by Barbosa [20, 18, 19], Lin [79, 78], and others [117, 46], it can be easily described using statistical modeling. In our approach, we focused on three packet features, namely, packet size, packet inter-arrival time, and packet direction. Using these features, we create a communication profile that describes statistical distribution of these features. We observe inter-arrival times Δt of incoming packets for each direction within a time window. For more accurate modeling, we add inter-arrival times distribution to the statistical model so that we split observed packets into several regions based on Δt values. Then we create a statistical model each region. Using this modeling, we are able to detect common ICS anomalies as communication loss, DoS attack, rogue device, scanning and switching attacks. By careful observing in which region the anomaly appeared, we not only detect the anomaly but also identify the possible cause of anomaly which is very important for security monitoring and management.

The last chapter concludes this work and discusses future research direction in the area of security of ICS communication.

1.2 Contribution

This document presents research results that were done by the author during 2016-2021, mostly in frame of projects IRONSTONE (2016-2019) and Bonnet (2019-2022), see below. The main contribution of the proposed work includes the following achievements:

- Definition of ICS flow. Design of ICS flow records for common ICS communication protocols in smart grid based on extended IPFIX flow records. This proposal was implemented into a commercial IPFIX monitoring probe by Flowmon Networks, Ltd within the IRONSTONE project (2016-2019). This is described in Chapter 3.

- Application of simple statistics and rule-based reasoning on extended ICS flow records obtained from the monitoring probe. The work was presented on the 6th International Symposium for ICS & SCADA Cyber Security Research in 2019 (ICS-CSR 2019) [88]. Extended version of the paper was published in Journal of Information Security and Application in 2020 (JISA, Q2) [90]. This is included in Section 3.7.

- Proposal of automata-based approach for representing ICS communication. Extraction and modeling IEC 104 features using Prefix Trees (PTs) and Deterministic Probabilistic Automata (DPAs). Evaluation of the method. The main results were presented in 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM 2021, Core B) [87]. The method is being implemented into the anomaly detection module of Flowmon Network's solution within the Bonnet project (2019-2022). The method is described in Chapter 4.
- Proposal of statistical model of inter-arrival times distribution, experiments with different features, evaluation. The paper with the method and experimental results was accepted to the 17th International Conference on Network and Service Management (CNSM 2021, Core B). The method is going to be implemented into the anomaly detection module of Flowmon Network's solution within the Bonnet project (2019-2020). Description of the method and our results is in Chapter 5.

1.3 Acknowledgment

The presented work was created as a part of the following research projects:

- Security monitoring of ICS communication in the smart grid (Bonnet), MV CR, VI20192022138, 2019-2022.
- Application of AI methods to cyber security and control systems, BUT, FIT-S-20-6293, 2020-2022.
- Traffic Analysis and Security Operations for ICS/SCADA (TRACTOR), TACR National Competence Center 1, TN01000077/02, 2019-2021.
- IRONSTONE - IoT monitoring and forensics, TACR Delta, TF03000029, 2016-2019.

The author would like to thank his colleagues Ondřej Ryšavý, Vojtěch Havlena, Ivana Burgetová, and Lukáš Holík from Brno University of Technology for their contribution on the presented research.

Chapter 2

Smart Grid Communication: Threats and Security

In this chapter, a standard architecture of the smart grid and its communication will be introduced. We mention typical properties of smart grid communication that are important for security monitoring and anomaly detection. Then we give an overview of cyber security threats for ICS and SCADA systems as identified by standards NIST SP 800-82 [113], NISTIR 7628r1 [36] and other works [122, 72, 98, 81]. We show typical cyber attack scenarios on ICS communication and discuss security recommendations for enhancing security of smart grid communication which is the main objective of this work.

2.1 Industrial Control Systems

With the progress of digitization, industrial automation, and intelligent control of industrial processes as proposed by the Industry 4.0 initiative [111], *Industrial Control System* (ICS) communication plays an essential role in monitoring and controlling devices, processes and events. The term ICS includes a wide range of control systems like Process Control System (PCS), Distributed Control System (DCS), *Supervisory Control And Data Acquisition* (SCADA) system, Substation Automation System (SAS), Safety Instrumented System (SIS), and others [72].

The main purpose of industrial control systems is to gather real-time data, realize device automation, and supervise the system. The ICS system is not limited to power industry and energy distribution but it is also deployed in factory automation, oil and gas industry, water distribution, transportation, air control, etc.

Industrial control systems belong to the broad category of *Operational Technology* (OT). OT generally includes the hardware and software used in industry for monitoring and control of physical devices, processes, and events. The term operational technology is used to demonstrate technological and functional differences to traditional *Information Technologies* (IT) which were originally designed for administrative tasks and communication over the Internet.

2.1.1 IT and OT Networks

Until recently information technologies (IT) and operational technologies (OT) lived in separate worlds. IT supported connections over the Internet, OT monitored and controlled devices and processes in the industrial environment. Traditionally, OT has used dedicated networks with specialized communication protocols. OT devices used to run separately from the IT networks, see Figure 2.1.

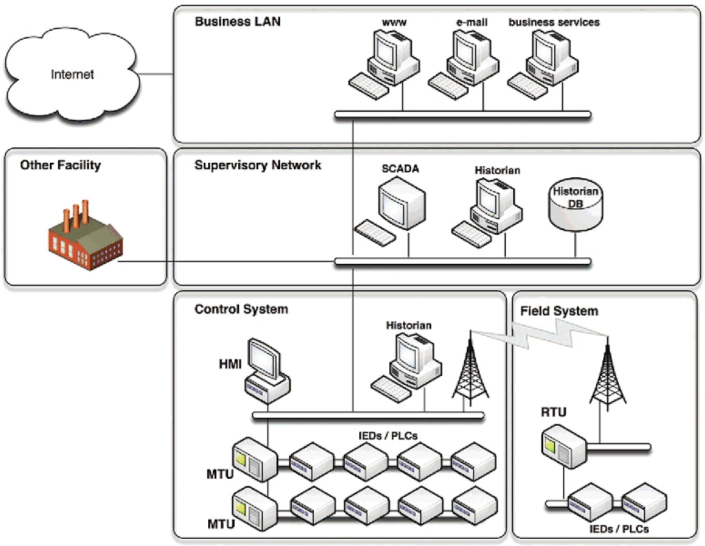


Figure 2.1: Separation of business and industrial networks [72]

With the rise of the Internet and expansion of Ethernet and TCP/IP as standard technologies for IT communication, OT has also started to adopt these technologies for OT domain. A common example is the Ethernet that replaced serial industrial buses, or the IP protocol for transporting control communication in the industrial environment. The convergence of IT and OT networks increased accessibility of OT devices through the remote control and monitoring over WAN networks. This resulted in major security concerns, particularly because many OT systems and devices were never envisioned to run on a shared, open standard infrastructure with built-in security. The following items highlights main differences between IT and OT networks [51]:

- OT networks run 24x7 and are part of the critical infrastructure. Their disruption directly impact business, energy (electricity, gas, water) supply, etc.
- OT top priorities are (1) availability, (2) integrity, and (3) security while IT prioritizes (1) security over (2) integrity and (3) availability.
- OT networks transmit monitoring, control, and supervisory data only while IT traffic also includes voice, video, transactional, or bulk data.

- OT security focuses on controlling physical access to devices. IT security includes user and devices authentication, data integrity and data encryption.
- OT networks were often isolated and used proprietary or ISO/IEC protocols. Some data are transmitted directly over L1, others require the full OSI stack.
- OT communication is mostly insecure. Protocols do not implement access control, authentication, authorization, or encryption by design. When transmitted over WAN networks, SSL/TLS or IPSec security are applied.

2.2 Smart Grid Network Architecture

The smart grid presents an ICS domain in energy industry. Term *smart grid* is generally used for the *next-generation power system that integrates high-speed and two-ways communication technologies into millions of power equipment to establish a dynamic and interactive infrastructure with new energy management capabilities, such as control, monitoring, load management system, power distribution, advanced metering infrastructure (AMI)* [72, 126].

As seen in Figure 2.2, the smart grid communication includes the backbone network and local area networks (LANs) [122]. The backbone network consists of gateways and high-bandwidth routers that interconnect LANs and forward messages across a variety of domains in the smart grid. SCADA system of the smart grid provides monitoring and control functions across the operations, transmission, and distribution domains. The LAN is used for intra-domain communication. It includes end nodes, for example meters, sensors or intelligent electronic devices (IEDs) installed on the power infrastructure.

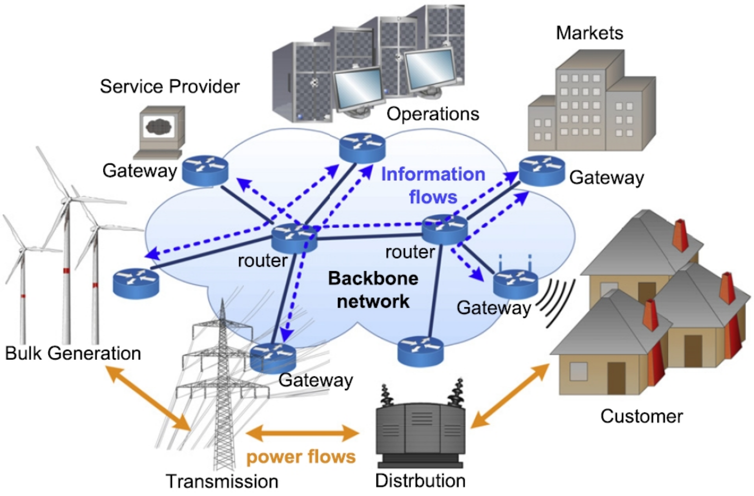


Figure 2.2: The network architecture in the smart grid [122]

On one hand, integration of advanced computing and communication enhances efficiency and reliability of future power systems, energy distribution, metering, remote control, and monitoring. On the other hand, convergence of IT and OT technologies in smart grid infrastructure, especially interconnection of industrial and TCP/IP networks, reveals new vulnerabilities associated with easier access to control bus and highlights a lack of security features of industrial protocols and end devices.

As mentioned above, security of smart grid systems is critical. With millions of electronic devices inter-connected via communication networks, network intrusions may lead to a variety of severe consequences in the smart grid such as disruption of operation or even a massive blackout as showed the attacks on Ukraine's power system in December 2015 [76] and 2016 [17, 30]. In these cases, communication system of Ukrainian power grid was infected by a malware which started to sent authorized commands to switch breakers and caused disruption of the service with the following collapse of the whole control system.

2.2.1 Types of Communication in the Smart Grid

The smart grid is a network that integrates information and communication technologies with the power-delivery infrastructure, enabling two-way energy and communication flows [36]. It provides three major functions [51]: (i) *power generation* in nuclear, coal, power, or wind plants where the electricity gets produced, (ii) *power transmission* that takes high-voltage electrical power over long distances to substations in the service area, and (iii) *power distribution* that delivers energy from the substation to homes or businesses. Management of these main operations include metering, distribution management, billing, or demand response, see Figure 2.3. Smart grid control communication covers two application domains [122]:

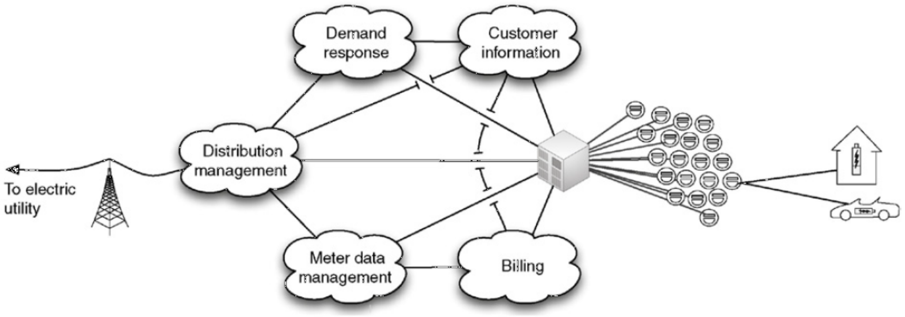


Figure 2.3: Smart grid deployment [72]

- *Distribution and Transmission Operation.* Power distribution and transmission operation systems are responsible for power delivery between power generators (power plants) and customers from the perspective of control.

These systems consists of millions of devices like Remote Terminal Units (RTUs), Intelligent Electronic Devices (IEDs), Programmable Logic Controllers (PLCs), and others, that are interconnected with the SCADA server for centralized management.

SCADA communication collects measurements and status data and sends control commands to switching devices (e.g., circuit breakers). Based on the collected data, a management system provides analytical tools for operators to determine the system state and take appropriate actions.

An important part of the power grid comprises substations. All devices in a substation are controlled, protected and monitored by substation automation system (SAS) that collects information from the power equipment and performs actions on it. SAS uses Ethernet-based communication network. Communication in the system is time-critical. Standard IEC 61850 [8] defines high-speed communication protocols for substation automation facilities. The standard describes three transmission protocols: Generic Object Oriented Substation Event (Goose), Manufacturing Message Specification (MMS) and Sampled Measured Values (SMV). GOOSE transmits messages from protective IEDs to circuit breakers. Measurement quantities, e.g., current or voltage values, are sent from measuring units to IEDs by SMV protocol. The MMS protocol provides the exchange of system data and control commands between a user interface and IEDs, see Figure 2.4.

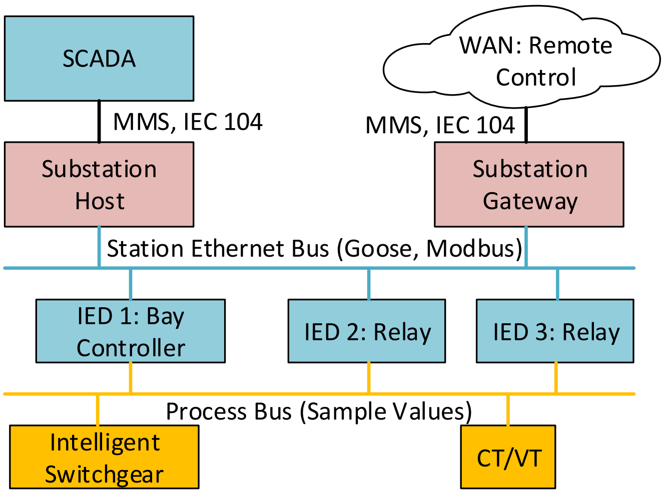


Figure 2.4: An example of SCADA network in the substation

Besides IEC 61850 protocols, smart grid network may deploy protocols like Modbus, DNP3 or IEC 60870-5-101/104 [35] that are also used for controlling power grid devices.

- *Advanced Metering Infrastructure (AMI) and Home-Area Networks (HAN).* The AMI system connects each customer's home-area network for scheduled energy management. Communication here is primarily for interaction with customers and utilities. It includes information exchange between smart meters and the utility center, such as reading and maintenance. Message delivery in the AMI network is non-time critical and availability is less important than integrity and confidentiality. Communication network of AMI is formed by smart meters, local data aggregators and management system.

  Legacy devices for remote metering known as Automatic Meter Reading (AMR) supported only one-way communication and were not able to deliver management and control messages. They communicated through serial RS-232 interface or infrared (IF) connection [70]. On lower layers, communication to the AMR server was transmitted via power line carrier, cellular networks (GSM, GPRS), telephone lines PSTN, or short range radio frequency (RF) networks (Bluetooth, WiFi, ZigBee, wireless M-Bus or WiMax) [120]. A new generation of meters known as smart meters is equipped with advanced hardware and software capabilities and communicates over TCP/IP via high-level protocols like DLMS/COSEM [11, 1].

2.2.2 Common Properties of ICS Communication

ICS and SCADA networks provide control and monitoring operations over the network infrastructure. SCADA servers continuously poll RTU and IED devices to get their status and variables that are used to monitor industrial processes, eventually send control commands. Alerts are generated in case of unusual events when human intervention is required. Many researchers examined control communication in ICS networks, observed its properties, and identified typical communication patterns [18, 19, 117, 78, 69]. A following list comprises major features of ICS communication in the smart grid that are important for modeling ICS communication and anomaly detection.

- *Delays.* Power communication is not designed to provide high throughput services as required by Internet connection but to ensure reliable and secure transmissions, in addition with real-time message delivery. Hence, latency is much more important than the throughput.

  Internet services often define different requirements on transmission delays. Real-time communication like VoIP requires delays below 150 ms while service with human interaction, e.g., ssh, web, or ftp can experience longer delays. The smart grid also features a wide range of delays from milliseconds to minutes. For example, messages for trip protection in substations have the delay constraint of 3 ms [13]. Thus, time-critical messages are sometimes passed from the application layer directly to the MAC layer to avoid redundant processing, e.g., in GOOSE, GSE, or SMV communication [8]. An important task of SCADA networks is to guarantee time response [73].

- *Periodicity.* In power networks, a large amount of traffic flows shows periodic behavior [122] because they are initiated by pre-configured electronic devices that run constantly without interruption for months. Unlike Internet traffic, there is no much human interaction involved in communication and therefore a traffic pattern is more predictable. We can observe several phases of power system communication, namely (i) *initialization* when a connection is established and values of all available variables are being set, (ii) *normal transmission* which mostly comprises periodic reading of variable values and checking the status of a device, and (iii) *configuration* that includes writing values or issuing remote commands.

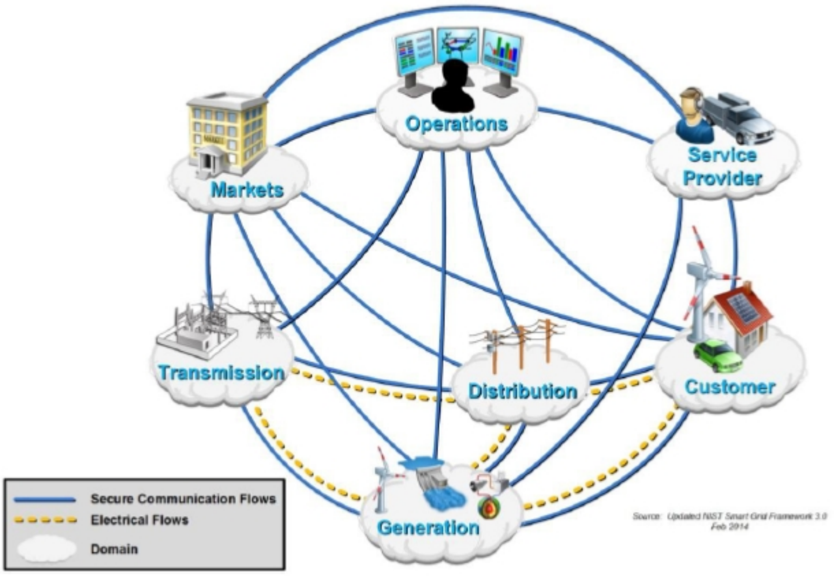
  Common cyber attacks like intrusions, network scans, or DoS attacks disturb traffic periodicity. Periodicity checking requires to learn the periodicity at first. During periodicity learning we extract two characteristics of network flows: the period (frequency) and the size of the periodic bursts which are used to compute a periodogram [19]. Anomaly detection consists of measuring distance between periodograms of learned flows and the periodogram of a running flow.

- *Constant throughput.* Barbosa et al. [18] also examined time series of throughput (packets/sec) in SCADA communication that showed constant shape over long periods of time often called *a baseline*. This feature is reflected in statistical anomaly detection, see Chapter 5.3.

  Unlike traditional IP network, SCADA communication does not show diurnal patterns as proved by [20]. The reason is that SCADA communication is mostly initiated by an automated process while Internet communication pattern reflects individuals' activity.

- *Communication Model.* In legacy power grids, the most commonly used communication model is a *one-way communication* where electronic devices report their readings to the control center. In contrast, the smart grid enables a *two-way communication model*: top-down (center to a device, control direction) or bottom-up (device to center, monitoring direction). The smart grid also supports *peer-to-peer communication*, e.g., IEC 104 communication. The type of ICS communication model is considered in statistical modeling (Section 5.2.4) where we define *master-oriented* communication profile that corresponds to one-way transmission, e.g., for GOOSE publish-subscribe mechanisms, and *peer-to-peer oriented* profile that is applied to peer-to-peer protocols like IEC 104.

- *Monitoring.* The smart grid network spans over a large geographical area. It uses hierarchical topology of independent systems like substation automation, smart meters, distribution system control, and others. The NIST *Framework and Roadmap* [36] identifies seven operation domains within the smart grid: Transmission, Distribution, Operations, Generation, Markets, Customer, and Service Provider, see Figure 2.5.



Source: Updated NIST Smart Grid Framework 3.0
Feb 2014

Figure 2.5: Smart grid domains [36]

All these domains need to transmit, store, and process the information needed within the smart grid. It is almost impossible to secure every part or node against physical attack. Therefore, the communication network needs to continuously perform profiling, testing and comparison of the network traffic status so that we can detect and identify abnormal behavior caused by malfunctioning or security incidents. A solution how to increase security monitoring of smart grid communication is addressed in Chapter 3.

- *Network stability.* The smart grid network is composed of devices which are permanently connected to the network either in on or off state. Thus, it is possible to track all addresses transmitted over a communication line and identify unknown or unexpected resources that appear in communication.

  ICS communication also exhibits a stable number of connections within a baseline that is almost unchanged for periods longer than a day [18]. When studying the graph of active connections of the smart grid network during a given period, there are not many changes with historical records.

  Stability feature includes the stable number of connected devices for longer periods, stability of established connections between communicating nodes, stability of transmitted packets/bytes between nodes, a stable or even fixed range of communicating protocols that appear on the line, etc. Due to network stability we can apply whitelisting [22, 72] as a powerful and easy-to-implement technique for protecting smart grid resources.

Stability indicators like the number of active devices, their addresses, the number of active connections, the number of transmitted packets and bytes, etc., can be obtained through enhanced IPFIX monitoring as proposed in Section 3.3. By observing and analyzing these indicators, we can detect various security incidents, connection loss, or device failures.

- *Communication Stability.* Direct consequence of network stability is communication stability. As demonstrated by previous research and also our experiments, ICS devices with pre-configured services and applications communicate with each other using a stable set of messages that correspond to the type of a device, its configuration, and used ICS protocol.

  By monitoring normal ICS communication we can learn typical communication sequences and create a profile of device-to-device communication using a formal language. Our experiments in Chapter 4 show that the set of exchanged messages is small and can be efficiently modeled using finite automata.

The above mentioned properties of smart grid communication are essential for security monitoring and anomaly detection which is the core of this work. The first task of the process is to observe ICS communication in real time and extract values of interest from ICS packets that represent network activity. The following task is to create a model of normal communication using the extracted values. Such model represents the baseline. Whenever the communication on the network significantly differs from the learned model, we indicate an anomaly. During detection phase we can apply various detection techniques in order to reveal potential security threats.

The next section describes typical cyber security threats against smart grid communication with specific focus on attacks against Ukrainian smart grid that happened in 2015 and 2016.

2.3 Cyber Security Threats

Cyber security of ICS protocols is still insufficient in comparison with Internet or mobile networks because design of many ICS protocols does not include encryption, authentication is often implemented with a pre-configured password transmitted in an open form, and a functional scheme for key management of ICS end devices is missing [125].

This section discusses security issues of smart grid communication. First, we present requirements for smart grid cyber security and overview common threats against ICS systems. Then we briefly mention data security standard IEC 62351 [61] for smart grid communication and data exchange. Finally, we give an overview of past cyber attacks against ICS and SCADA systems and explain typical attack scenario using examples of two attacks against Ukrainian power grid.

2.3.1 Requirements on Cyber Security

Smart grid communication networks belong to critical infrastructure systems. NIST *Guidelines for Smart Grid Cybersecurity* [36] defines three high-level security objectives for the smart grid, namely *confidentiality*, *integrity*, and *availability* (CI&A) which are similar to IT networks security requirements. In addition, the document mentions *access control*, *configuration management*, *identification and authentication*, *audit and accountability*, *continuity of operations*, *incident response*, and others. Similarly, standard IEC 612351 for security of smart grid communication [60] lists four main security requirements: *confidentiality*, *integrity*, *availability*, and *non-repudiation*.

- *Authentication, authorization, access control.*

  The smart grid network infrastructure incorporates millions of electronic devices and users. Authentication verifies the identity of a user or a device while authorization grants legitimate access to a resource in the smart grid. Authentication and authorization require secure management of credentials that are used to prove an identity of an entity requesting a service, and also to check permission to use the service. Authentication and authorization shall be implemented both on communication level (protocols) and also on end devices. Unfortunately, many ICS protocols do not support authentication or authorization, or these features are optional and not implemented at devices.

- *Integrity, confidentiality.*

  Integrity means the protection against modification or destruction of transmitted data. Confidentiality protects data against non-authorized access and reading, which is important for billing and smart metering. Message delivery in ICS systems is mostly time-critical and requires fast packet generation and processing while implementation of security adds an extra time for computing a MD5 hash or encrypting the packet. Thus, optimal balance between fast packet delivery and sufficient security must be found.

  In IT networks, confidentiality is mainly implemented using secure connection via VPNs, IPSec, or SSL/TLS. These strategies can be also applied to OT environment, especially for WAN communication. Direct application of IT security techniques is limited due to time-critical processing [95].

- *Availability.*

  Many processes in smart grid networks are continuous in nature. Unexpected outage of a system that controls industrial processes is not acceptable. Using IT strategies such as rebooting is not possible due to the impact on the system [113]. Thus, redundant components and backup communication lines are required. The IEC 61850-90-4 standard for substation protection considers Rapid Spanning Tree Protocol (RSTP), Parallel Redundancy Protocol

> (PRP) and High-availability Seamless Redundancy (HSR) as recommended solution for IEC 61850-8-1 and IEC 61850-9-2 communication [95].
>
> Attacks against availability include Denial of Service (DoS) attacks on Layers 1 or 2, e.g, jamming attack, SYN or Smurf attacks on Layer 3, or DoS attack on Layer 7.

When talking about smart grid cyber security, we usually focus on a single part of smart grid architecture. Thus, we talk about cyber security of the power substation system, SCADA system, AMI network, PMU synchronization, etc. However, above mentioned requirements shall be implemented on every single system of the smart grid. Naturally, availability would be more important for smart metering while authentication and authorization is important for SCADA control and monitoring transmissions. Nevertheless, all above mentioned requirements should be carefully considered for each part of the smart grid system.

2.3.2 Common Threats to Smart Grid Communication

In this part, we overview main network security threat categories according to [72, 122, 98, 81, 60, 125]. Nevertheless, many cyber attacks exploit more than a single vulnerability of the target, thus we talk about *blended attacks* [72]. It is important to analyze features of a possible attack before starting to think about detection and prevention.

2.3.2.1 Network Scanning

Network scanning is considered as a preliminary phase of an attack. Before the attack is launched against the target, the attacker gathers information about victim's network: its topology, addressing scheme, running protocols and services, installed software, etc. This phase is also known as *reconnaissance* and is performed either passively by intercepting passing traffic, or actively by sending special requests into a network with the objective of discovering available services and devices.

Network scanning is very easy to execute in IT networks, so is is often ignored by IT security experts. In contrast to IT domain, OT systems are less open to outside communication and the traffic is more predictable because of automatic processes. Thus, network scanning of ICS networks is considered as a significant signal of starting attack that should be detected and thoroughly examined in order to prevent following hostile activities.

Past ICS cyber attacks show that network scanning attacks of ICS networks is usually launched from a compromised host inside the network. Thus, traffic filtering on the perimeter of the ICS network cannot detect and prevent network scanning and advanced detection methods like behavior-based anomaly detection and communication pattern analysis [117] are needed.

2.3.2.2 Denial of Service (DoS)

The primary objective of a DoS attack is to delay, block, or corrupt communication in the smart grid, so it violates system availability. DoS attack can be performed on various layers of communication model:

- *Physical layer.* When performed on physical layer, it is often called *channel jamming*. Channel jamming is usually launched against wireless connection in substations where causes a wide range of damages [82]. For detection, a signal-based detector can measure the received signal and detect the presence of jamming. If the signal strength is larger than a threshold, the detector raises alarm.

- *Data link layer.* On Layer 2, a malicious node can start sending excessive amount of broadcast messages (broadcast storm) which disrupt the whole LAN communication.

- *IP layer.* Similar attack can be launched on Layer 3, e.g., a ping attack against IEC 61850 substation as demonstrated by [98].

- *Application layer.* On application layer, the DoS attack can send an enormous amount of packets that a target host is unable to process. This was demonstrated on real ICS network where 100,000 unsolicited DNP3 event messages were sent to the DNP3 master from a compromised IED [68].

DoS attacks initiated from the outside of the ICS system are usually filtered out by the perimeter IDS or firewall. In this work we focus on security monitoring of internal ICS systems. External attacks are not the subject of our research.

To detect DoS attacks on Layers 2 to 7, we can apply similar methods as in IT networks. Based on monitoring statistics we can detect a high number of failed transmission, increased traffic load and higher packet transmission ratio [122]. Another mitigation technique includes policy-based routing where a sender must obtain authorization to send significant amount of traffic, stateless dynamic packet filtering, lightweight authentication and authorization, etc. [36].

Since the smart grid communication exhibits two major predictable directional information flows, i.e., bottom-up (monitoring direction) and top-down (control direction), we can easily implement firewall rules to filter undesired or suspicious traffic flows [122].

2.3.2.3 Man-in-the-Middle (MITM)

MITM attack can be launched either from a compromised host in the ICS network or from an unauthorized (rogue) device that was connected to the ICS network. This kind of attack is especially dangerous for smart grid networks where outstations located in remote places can be a subject of physical attack. When an attacker gains access to the transmission medium, it can launch a spoofing attack where it masquerades itself as another device. An attacker can also redirect Layer 2 or Layer 3 communication to itself via spoofed ARP or IP packets.

Spoofed devices can be later misused to eavesdrop communication, launch a DDoS attack, or send forged commands into the ICS system in order to turn down critical industrial processes.

MITM attack can be prevented by a proper usage of authentication and by ensuring integrity checks of transmitted data as recommended by standard IEC 62351 [61]. Unfortunately, authentication is rarely implemented in ICS networks. As shown further, MITM attacks can be successfully identified by proposed detection techniques as mentioned in Chapters 4 and 5.

2.3.2.4 Data Interception and Manipulation

Data interception can be a part of MITM attack or it can be caused by wiretapping the transmission line. This attacks is directed against data integrity and confidentiality with possible attempts to access or/and modify data. The target can be customers' information (e.g., pricing, billing, account balance), status values of the system (e.g., voltage reading, device status), or other information.

One of the most discussed variant of data interception in smart grid is a *false data injection attack* (FDIA) [80, 77, 124] which targets Advanced Metering Infrastructure (AMI) communication. This attack is focused against state estimation in electric power grids. State estimation is a process of estimating unknown state variables in a power grid based on the meter measurements. The output of state estimation is typically used in contingency analysis, which controls the power grid components, e.g., the increase of the yield of a power generator [80]. Malicious measurements injected by an attacker can mislead the state estimation process [77], manipulate the market price information [124] or cause load redistribution [127] which can lead the system into a false operating state.

Data interception can be generally prevented by encrypting transmitted data.

2.3.3 Limitation of Security Standard IEC 62351

Standard series IEC 62351 [61] defines data security in the power system. It addresses common cyber security threats mentioned in the previous sections and includes recommendations for security improvements of smart grid communication protocols IEC 61850 (MMS, GOOSE, SMV), IEC 60870-5 (IEC 104), and others. Specification covers authentication through digital signatures, prevention of eavesdropping, anti-replay attacks, spoofing and other threats. The standard consists of several parts, see Table 2.1.

However, there are signification objections concerning feasibility of implementation of the standard in smart grid devices. In addition, not all parts of the standard are properly designed and may cause additional vulnerabilities [110, 114, 122]. The following objections are raised:

Part	Description
IEC 62351-1:2007	Introduction to security issues
IEC 62351-2:2008	Glossary of terms
IEC 62351-3:2014	Profiles including TCP/IP
IEC 62351-4:2007	Profiles including MMS
IEC 62351-5:2013	Security for IEC 60870-5 and derivatives
IEC 62351-6:2007	Security for IEC 61850
IEC 62351-7:2017	Network and System Management data object models
IEC 62351-8:2011	Role-based access control
IEC 62351-9:2017	Cyber security key management for power system equipment
IEC 62351-10:2012	Security architecture guidelines
IEC 62351-11:2016	Security for XML documents
IEC 62351-12:2016	Resilience and security recommendations for power systems with distributed energy resources cyber-physical systems
IEC 62351-13:2016	Guidelines on security topics to be covered in standards and specifications
IEC 62351-90-1:2018	Guidelines for handling role-based access control in power systems

Table 2.1: IEC 62351 standard for smart grid security

- The standard does not provide a way of adequate defense of compromised devices and these devices will be still recognized as legitimate by other nodes in the system.
- Due to backward compatibility, IED devices offer both secure and insecure communication. Thus, an attacker can choose to use insecure channel to bypass authentication or encryption requirements.
- Application level security (A-profile) does not cover message integrity and confidentiality of all transmitted messages but only during initial phase when connection is established. This means that an attacker can forge or modify PDUs exchanged between two devices.
- Security enhancements of SCADA device can be misused by a DoS attack targeted on limited CPU and memory resources of SCADA nodes.
- The Standard specifies authentication for the communication between two stations, but at the same time it allows the authentication state to be shared between stations by improperly authenticated messages. This may cause

invalidation of session keys or forcing discarding legitimate messages with authentication.

- Some applications require response times of 4 ms. The standard does not recommend encryption for these applications while not giving any other form of protection [95].

- Key management is a great challenge for the SCADA system where a substation communicates with hundred IEDs. Due to the lack of resources and low latency requirements in SCADA networks, it is unfeasible to use traditional key management schemes proposed by the standard. Open issues include key generation in the device, key exchange, distribution, or revocation in SCADA networks [47].

- GOOSE/SMV protocol security does not cover anti-replay attack protection as demonstrated by [114].

Based on aforementioned description, we have identified following challenges related to cyber security of smart grid communication. These challenges are later addressed by anomaly detection techniques proposed in the following chapters.

- Smart grid communication protocols are not secure enough to provide authentication, integrity and confidentiality. When a device in the smart grid network is compromised, it can be misused to intercept communication, modify packets or inject data on the communication line. These forged packets will be considered as legitimate since there is no way how to verify their origin or integrity on the level of communication protocol.

- Smart grid networks are vulnerable to DoS attacks. Industrial IDS systems can filter attacks initiated from the outside of a substation network, however, filtering is powerless if an attack is launched from the inside of the network. Smart grid communication (e.g., GOOSE communication) is time critical, thus, even moderate-range DoS can disrupt operation of the substation.

- A high level penetration of smart meters increases a possibility that intruder may access the AMI network from a node installed in the public area, e.g., a smart meter and local data collector [115]. Compromising a host may lead to energy theft, false data injection, or leakage of the customer information.

- Current cyber security protection concentrates on smart grid network perimeter where firewalls and IDS systems are used to identify possible attacks and filter incoming traffic. These devices lack a *clear visibility of ICS communication* that may reveal security incidents or technical malfunction of critical systems.

As mentioned above, smart grid infrastructure is susceptible to a wide range of different cyber attacks due to its distributed nature, unprotected transmission protocols, and remote control features. Today, cyber security in the smart grid is mostly provided using (i) network segmentation via VLANs that separates ICS communication and non-production traffic, (ii) firewalls that filter incoming and outgoing communication, and (iii) IDS devices that provide advanced analysis of network flows and detect anomalies based on signatures or behavior analysis. In addition, (iv) ICS/SCADA honeypots, (v) probes or analysis modules are being deployed to protect ICS network against cyber attack [67].

The performance of firewalls or IDS systems mostly rely on predefined rules or a set of signatures of known attacks. To be able to identify a new type attack (also called zero-day attack), advanced techniques like anomaly detection are needed.

2.4 Cyber Attacks against ICS Systems

Cyber attacks against industrial control systems are not just a theory. One of the first documented attack happened during the Cold War in 1982 when intruders from Central Intelligence Agency (CIA) planted a Trojan horse in the SCADA system that controlled the Trans-siberian pipeline. The software reset pump speeds and valve settings and produced so high pressure that caused an explosion [93, 105].

Later, plenty of attacks on SCADA systems were performed. Some of them were caused by attackers who broke to the SCADA system from inside by exploiting L1 or L2 communication, e.g., knocking out of Worcester air traffic system in 1997, Maroochy Shire sewage spill in 2000 [113]. Other attacks started by malware infection via phishing mails that caused infection of a device inside the SCADA network and opened back-doors for an attacker. Following that, DoS attacks were launched against SCADA system, for example, worm Slammer attacked SCADA system of David-Bess nuclear power plant in 2003 and caused a crash and unavailability of Safety Parameter Display System (SPDS), Sobig virus that caused shutting down of CSX train signaling system in 2003, Zotob worm knocked offline 13 of Daimler-Chrysler's manufacturing plants in 2005, see Table 2.2.

Besides DoS attacks focused on putting parts of the operational technologies out of service, there were also sophisticated attacks directed to seize a control over technological process. Such attacks are also called *Advanced Persistent Threats* (APTs) [38] because they represent continuous hacking activities against a target. The most popular APTs are Stuxnet worm that destroyed Iranian centrifuges by increasing and decreasing their speed and pressure beyond normal levels in 2010 [113], and attacks against Ukrainian power grid, see Sections 2.4.1 and 2.4.2 .

2.4.1 BlackEnergy Attack on the Ukrainian Power Grid (2015)

On December 23, 2015, Ukrainian power companies experienced unscheduled power outages for a few hours impacting a large number of customers in the Ivano-Frankivsk region in Ukraine (population around 1.4 million). There have also been reports of malware found in Ukrainian companies in a variety of critical infras-

Year	Attack	Place
1982	Explosion of Siberian gas pipeline caused by a trojan which reset pump speeds and valve settings	Soviet Union
1997	Knock out of Worcester air traffic control communication	MA, USA
2000	Maroochy shire sewage spill	Australia
2003	Crash of the Safety Parameter Display System in Davis-Bess nuclear power plant by Slammer worm	OH, USA
2003	Shutting down of CSX train signaling system by Sobig virus	FL, USA
2005	Zotob virus knocked 13 of Daimler-Chrysler's manufacturing plants	USA
2010	Stuxnet virus damaged Iranian centrifuges by increasing and decreasing their speed and pressure beyond normal levels	Iran
2014	Disrupted control system in German steel mill	Germany
2015	Power outage off Ukrainian power plant distribution caused by BlackEnergy malware	Ukraine
2016	Industroyer malware attack on Ukrainian power grid	Ukraine
2017	Cyber-espionage attack against aerospace and energy industry by APT33 group	USA
2019	Cyber attack against chemical giant Bayer	Germany
2019	Intrusion attack against U.S. Energy sector by KAMACITE group	USA
2019	Cyber attack against Kudankulam nuclear power plant	India
2020	Compromising supply chain of Solarwinds software used in industrial environment	USA
2021	Ransomware attack against oil distribution company Colonial Pipeline	USA

Table 2.2: Overview of cyber attacks against ICS systems

tructure sectors. Public reports indicate that the BlackEnergy (BE) malware was discovered on the companies' computer networks.

Power outages were caused by remote cyber intrusions at three regional electric power distribution companies (Oblenergos) impacting approximately 225,000 customers. While power has been restored, all the impacted Oblenergos continued to run under constrained operations. In addition, three other organizations, some from other critical infrastructure sectors, were also intruded upon but did not experience operational impacts [76].

The cyber-attack was reportedly synchronized and coordinated, probably following extensive reconnaissance of the victim networks. According to company personnel, the cyber-attacks at each company occurred within 30 minutes of each other and impacted multiple central and regional facilities. During the cyber-attacks, malicious remote operation of the breakers was conducted by multiple human attackers using either existing remote administration tools at the operating system level or remote industrial control system (ICS) client software via virtual private network (VPN) connections. The companies believe that the actors acquired legitimate credentials prior to the cyber-attack to facilitate remote access.

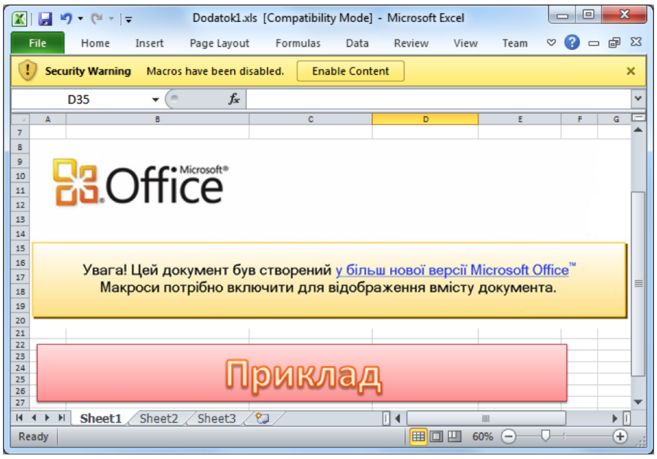
2.4.1.1 BlackEnergy Trojan

BlackEnergy is a trojan which dates back to 2007 [9]. Originally, it was designed as a toolkit for creating botnets for use in conducting DDoS attacks. Over time, the malware has evolved to support different plugins, which are used to extend its capabilities to provide necessary functions, depending on the purpose of an attack.

BlackEnergy 2 (BE2) [10] was first observed in 2010, used by advanced persistent threat (APT) group dubbed ”Sandworm” and was tailored to target industrial control systems (ICS) components–specifically human machine interfaces (HMIs) used in the industrial environment. Once the perpetrators gained access to the HMI they can conduct reconnaissance on the network.

The third iteration of BlackEnergy (BE3) emerged in 2014 featuring additional functionality such as support for proxy servers, espionage modules, and support for a range of operating systems and devices. The purpose of these plugins was mainly for network discovery, remote code execution, and for collecting data off the target’s hard drives. In addition, it contained a KillDisk component specifically designed to erase files and corrupt a system’s master boor record, effectively rendering the system inoperable.

The infection vector used in these attacks is Microsoft Office files containing malicious macros. In one case it was a MS Word document, see Figure 2.6, with a text trying to convince the victim to run the macro in the document. This is an example of using social engineering to attack the victim host instead of exploiting software vulnerabilities.



Security Warning Macros have been disabled. Enable Content

Увага! Цей документ був створений у більш нової версії Microsoft Office™
Макроси потрібно включити для відображення вмісту документа.

Figure 2.6: An example of a Word document with a hidden macro

In the second case, PowerPoint slideshow file (.ppsx) was used to execute a BlackEnergy dropper. The PowerPoint package contained two embedded OLE objects with a remote path where resource is located. It is a feature of Microsoft PowerPoint to load these files, but it turned out to be a dangerous one, since the objects could be downloaded from an arbitrary untrustworthy network location and executed with none of the warning pop-ups. This happens due to exploitation of vulnerability in OLE objects in MS Office documents which allow remote attackers to execute arbitrary code, see CVE-2014-4114[1].

2.4.1.2 Attack Scenario

The attack scenario was simple: the target received a spear-phishing email that contained an attachment with a malicious document. The Ukrainian security company CyS Centrum published two screenshots of emails used in BlackEnergy campaigns, where the attackers spoofed the sender address to appear to be one belonging to Rada (the Ukrainian parliament). The document itself contained a text trying to convince the victim to run the macro in the document. If victims were successfully tricked, they ended up infected with BlackEnergy trojan.

In addition, the Win32/KillDisk malware was found on the infected system. As well as being able to delete system files to make the system unbootable—functionality typical for such destructive trojans—the KillDisk variant detected in the electricity distribution companies also appeared to contain some additional functionality specifically intended to sabotage industrial systems. The scheme of the attack from March to December 23, 2015 is depicted in Figure 2.7 using ICS Cyber Kill Chain methodology [16]. Detailed description of the attack is given below:

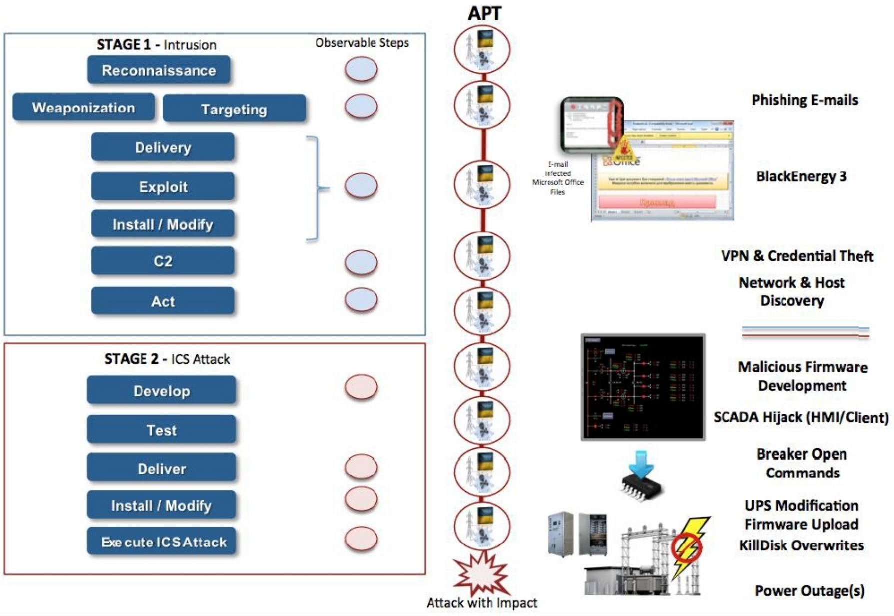


Figure 2.7: ICS Kill Chain Mapping Chart [76]

[1] Available at https://nvd.nist.gov/vuln/detail/CVE-2014-4114, [August 2018]

Reconnaissance. An analysis of the three impacted organizations showed [76] that they were particularly interesting targets due to the levels of automation in their distribution system; enabling the remote opening of breakers in a number of substations.

Weaponization and Targeting was focused on getting tools to break into the system. In this case, MS Office documents (Excel and Word) with embedding BlackEnergy 3 trojan were created.

Delivery, Exploit and Install. During these steps, the malicious documents were delivered via email to individuals in the administrative or IT network of the company. When these documents were opened, a popup was displayed to encourage users to enable macros in the documents. Enabling the macros allowed the malware to install BlackEnergy 3 on the victim system. Upon the install step, the BlackEnergy 3 malware connected to C&C IP addresses to enable remote communication with the malware and the infected system. From the analysis it seemed that attackers gained access more than six months prior to the attack. During that time the attackers gathered information about the system, credentials and discovered the system and extracted data necessary for the attack.

Develop and Test. In this stage the attackers learned how to interact with the local distribution management system using the native control and developed malicious firmware for the serial-to-Ethernet devices. It was also possible that attackers tested the malware prior to its deployment.

Deliver. Then the attackers used native software to deliver malware into the environment for direct interaction with the ICS components. They achieved this using existing remote administration tools on the operator workstations.

Install/Modify. In this stage, malicious software (customized KillDisk virus) was installed across the system.

Execute the ICS Attack. The final step was launched through HMIs in the SCADA environment to open the breakers. At least 27 substation were taken offline across the three energy companies, impacting roughly 225,000 customers for about 6 hours until manual operations could restore power. Simultaneously, the attackers uploaded the malicious firmware to the serial-to-Ethernet gateway devices. This ensured that even if the operator workstations were recovered, remote commands could not be issued to bring the substation back online. Thus the Ukrainian grid operators were without their SCADA environment, meaning that lost the ability for automated control, for upwards a year in some locations [41].

2.4.1.3 Mitigation

As seen from the above mentioned description, the attack included a range of various operations on the substation system that left traces on the targeting system and in network communication. Finding these traces and detecting a cyber threat in the beginning is the core of this document.

As reaction to this cyber attack against Ukrainian power grid, the US. Department of Homeland Security created a report[2] with following mitigation techniques:

- Implementation of information resources management best practices: trusted hardware and software, system patching, technology updates.
- Application Whitelisting (AWL): detection and prevention of attempts to execute malware uploaded by malicious actors.
- Isolation of ICS networks from any untrusted networks, especially the Internet.
- Unused ports should be locked down, all unused services turned off.
- Limit Remote Access functionality, use read only access for monitoring purposes.
- Strong multi-factor authentication should be used if possible.

Additional recommendations learned from this cyber attack are mentioned in E-ISAC report [76]. This report also recommends network security monitoring that continuously search the networked environment for anomalies. As we show in Chapter 3, the preliminary step for revealing malicious activities is visibility of network and system operations to the monitoring system. The second step is detection of anomalies that may include whitelisting, observation of untypical communication sequences, unusual number of transmitted packets, etc.

2.4.2 Industroyer Attack on the Ukrainian Power Grid (2016)

A week before Christmas 2016, hackers struck an electric transmission station of Ukrenergo north of the city of Kiev, blacking out a portion of the Ukrainian capital equivalent to a fifth of its total power capacity for an hour [50], see Figure 2.8. This would be the second such known hack of a Ukrainian power facility following a

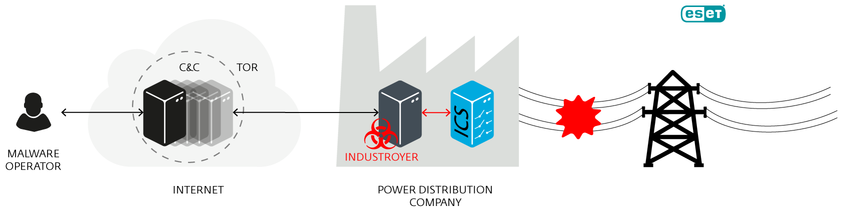


Figure 2.8: Industroyer operation [30]

[2] See https://ics-cert.us-cert.gov/alerts/IR-ALERT-H-16-056-01[April 2018]

massive December 2015 power outage affecting about 230,000 people, which was later blamed on the Russian government.

The attack occurred almost exactly one year after the previous outage, struck the Pivnichna substation outside the capital city Kiev, and cut power a few minutes before midnight local time December 17, leaving customers in part of Kiev and a surrounding area in the dark on a Saturday night. The outage lasted only an hour, and power was restored a little after 1 am.

Further analysis of the cyber attack was released by two cyber security companies, ESET and Dragos Inc., provided analysis of malware modules called *Industroyer* (ESET) or *CrashOverride* (Dragos), see [30, 41]. Comparing to the 2015 attacks, there are significant differences. The 2015 attack targeted three Ukrainian distribution entities causing distribution-level outages while damaging the utility's SCADA systems. The 2016 attack occurred at the transmission-level targeting a regional SCADA system generally focused on a single 330 kV-to-110 kV-to-10 kV substation [17].

2.4.2.1 Industroyer malware

The attackers employed trojan called Industroyer that is a sophisticated malware designed to disrupt operation of ICS systems, specifically in substations [30].

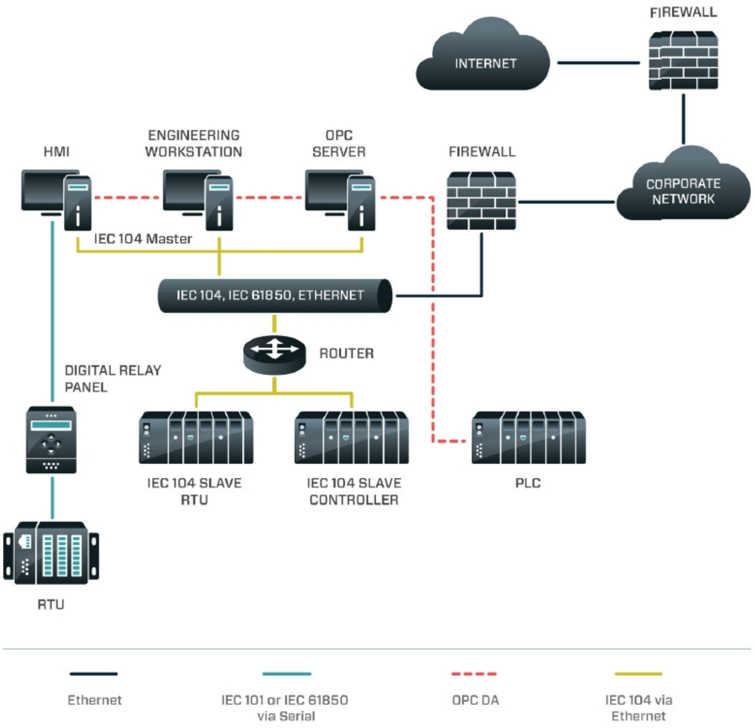


Figure 2.9: Model of power grid hit by Industroyer malware [41]

Industroyer is a particularly dangerous threat, since it is capable of controlling electricity substation switches and circuit breakers directly. To do so, it uses industrial communication protocols used worldwide in power supply infrastructure, transportation control systems, and other critical infrastructure systems (such as water and gas). The four industrial control protocols are included in Industroyer [30]: IEC 104, IEC 101, MMS/GOOSE, and OPC DA, see Figure 2.9. In addition, Industroyer implements a DDoS attack against a particular family of protection relays—Siemens SIPROTEC.

The problem is that the above mentioned protocols were designed decades ago without security in mind. That means the attackers didn't need to be looking for protocol vulnerabilities; all they needed was to teach the malware "to speak" with those protocols.

Industroyer was designed as a modular malware, see Figure 2.10. It consists of the main backdoor used by attackers to manage the attack. This module installs and controls other components and connects to a remote C&C server to receive commands and to report to the attackers. Post-mortem analysis revealed that most of the IP addresses or remote servers were part of Tor network[3] which covered the real origin of the attack.

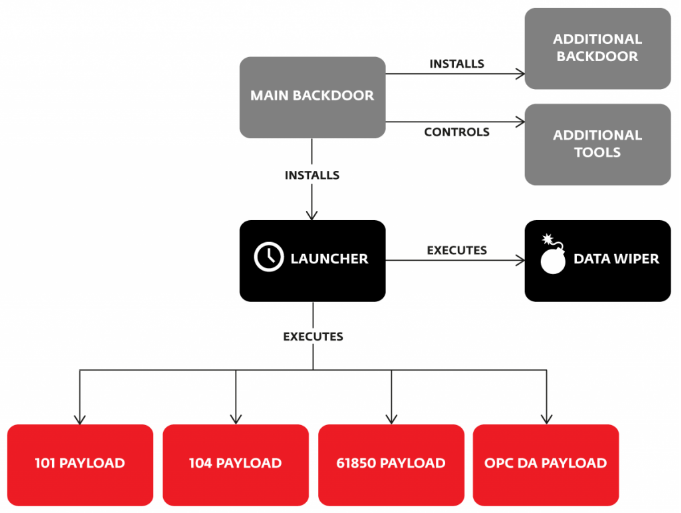


Figure 2.10: Schema of Win32/Industroyer components [30]

Similarly to the BlackEnergy 3, the Industroyer works in stages whose goals are at first mapping the network, and then figuring out and issuing commands that work with the specific industrial control devices.

Main Backdoor connects the malware to its remote C&C server using HTTPS and receiving commands from the attacker. All analyzed samples showed the hard-coded proxy address. This means, it was created to work only in one specific

[3] Tor (The Onion Router) is a network that enables anonymous communication.

organization. One interesting feature of this backdoor is that attackers can define a specific hour of the day when the backdoor is active, for example, outside working hours. Once connected to its remote C&C server, the main backdoor component sends the data in POST requests. The main backdoor component supports the following commands:

0. Execute a process.
1. Execute a process under a specific user account with supplied credentials.
2. Download a file from C&C server.
3. Copy a file.
4. Execute a shell command.
5. Execute a shell command under a specific user account with supplied credentials.
6. Quit.
7. Stop a service.
8. Stop a service under a specific user account with supplied credentials.
9. Start a service under a specific user account with supplied credentials.
10. Replace ”image path” registry value for a service.

Once the attackers obtained administrator privileges, they installed the backdoor to a more privilege level as a Windows service program. To do this they picked an existing, non-critical Windows service and replaced its `ImagePath` register value with the path of the new back-doors' binary.

Launcher was responsible for launching the payloads and the Data wiper component. It contained specific times and dates: 17th Dec 2016 and 20th Dec 2016. Once one of these dates was reached, the component created two threads:

- The first thread made attempts to load a payload DLL. The name of the payload DLL was supplied by the attackers via a command line parameters supplied in one of the main back-door's ”execute a shell command”.
- The second thread waited one or two hours and then attempted to load the Data wiper component.

101 payload component implemented communication between ICS system and Remote Terminal Unit (RTU) using a serial link. Configuration file of this malware component used several entries: a process name that was running on the victim machine, COM ports of the Windows station, Information Object Address (IOA) ranges, and the beginning and ending IOA values for the specified number of IOA ranges. The malware terminated the specified process and started to communicate with the specified device using `CreateFile`, `WriteFile` and `ReadFile` Windows API functions. It iterated through all IOAs in the defined range. For each IOA it constructed two `select and execute` packets, one with a single command SCO (`C_SC_NA_1`, type 45) and one with a double command DCO (`C_DC_NA_1`, type 46) sent to the RTU device. The attack itself had three stages:

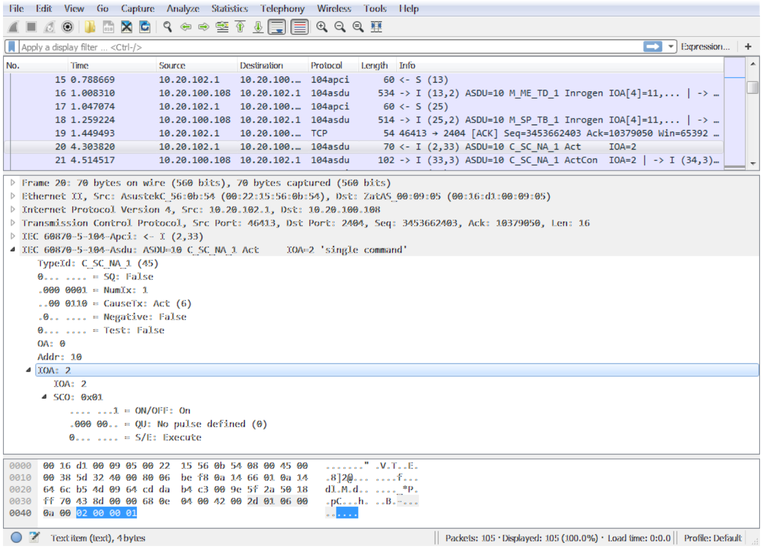
1. The malware attempted to switch IOAs to their OFF state.

2. It inverted IOA states to ON state.
3. It switched IOA states to OFF again.

104 payload component contained the IP address of the station, target port, ASDU address, switch value (on/off), change value (0/1), operation (iteration type for IOA: range, sequence, or shift). Once launched, it created a thread for each station section defined in the configuration file. In each thread it communicated with the specified IP address using IEC 104 protocol. Before the connection was made, the 104 component attempted to terminate the legitimate process that was responsible for IEC 104 communication.

Then, it connected to the the specified IP address and started to send packets with the ASDU address defined in the configuration where it interacted with an IOA using a single command type (SCO).

- `Range mode`. Using the `range mode` the attacker discovered all possible IOAs in the targeted device by enumerating all possible addresses. Once the range of valid IOAs was obtained, the malware iterated through the specified IOAs and sent `select and execute` packets to discover whether the IOA belonged to the SCO type, see Figure 2.11. Following that the malware started sending `select and execute` packets in an infinite loop where it flipped on/off state of the objects. If logging enabled, it wrote message `"Starting only success"` to the log.



```
No.   Time      Source          Destination    Protocol  Length Info
   15 0.788669  10.20.102.1     10.20.100.…    104apci       60 <- S (13)
   16 1.008310  10.20.100.108   10.20.102.1    104asdu      534 -> I (13,2) ASDU=10 M_ME_TD_1 Inrogen IOA[4]=11,... | -> …
   17 1.047074  10.20.102.1     10.20.100.…    104apci       60 <- S (25)
   18 1.259224  10.20.100.108   10.20.102.1    104asdu      514 -> I (25,2) ASDU=10 M_SP_TB_1 Inrogen IOA[4]=11,... | -> …
   19 1.449493  10.20.102.1     10.20.100.…    TCP           54 46413 → 2404 [ACK] Seq=3453662403 Ack=10379050 Win=65392 …
   20 4.303820  10.20.102.1     10.20.100.…    104asdu       70 <- I (2,33) ASDU=10 C_SC_NA_1 Act      IOA=2
   21 4.514517  10.20.100.108   10.20.102.1    104asdu      102 -> I (33,3) ASDU=10 C_SC_NA_1 ActCon   IOA=2 | -> I (34,3)…

▷ Frame 20: 70 bytes on wire (560 bits), 70 bytes captured (560 bits)
▷ Ethernet II, Src: AsustekC_56:0b:54 (00:22:15:56:0b:54), Dst: ZatAS_00:09:05 (00:16:d1:00:09:05)
▷ Internet Protocol Version 4, Src: 10.20.102.1, Dst: 10.20.100.108
▷ Transmission Control Protocol, Src Port: 46413, Dst Port: 2404, Seq: 3453662403, Ack: 10379050, Len: 16
▷ IEC 60870-5-104-Apci: <- I (2,33)
◢ IEC 60870-5-104-Asdu: ASDU=10 C_SC_NA_1 Act     IOA=2 'single command'
    TypeId: C_SC_NA_1 (45)
    0... .... = SQ: False
    .000 0001 = NumIx: 1
    ..00 0110 = CauseTx: Act (6)
    .0.. .... = Negative: False
    0... .... = Test: False
    OA: 0
    Addr: 10
  ◢ IOA: 2
      IOA: 2
    ◢ SCO: 0x01
        .... ...1 = ON/OFF: On
        .000 00.. = QU: No pulse defined (0)
        0... .... = S/E: Execute

0000  00 16 d1 00 09 05 00 22  15 56 0b 54 08 00 45 00   ......." .V.T..E.
0010  00 38 5d 32 40 00 80 06  be f8 0a 14 66 01 0a 14   .8]2@... ....f...
0020  64 6c b5 4d 09 64 cd da  b4 c3 00 9e 5f 2a 50 18   dl.M.d.. ...._*P.
0030  ff 70 43 8d 00 00 68 0e  04 00 42 00 2d 01 06 00   .pC...h. ..B.-...
0040  0a 00 02 00 00 01                                  ......

Text item (text), 4 bytes    Packets: 105 · Displayed: 105 (100.0%) · Load time: 0:0.0    Profile: Default
```

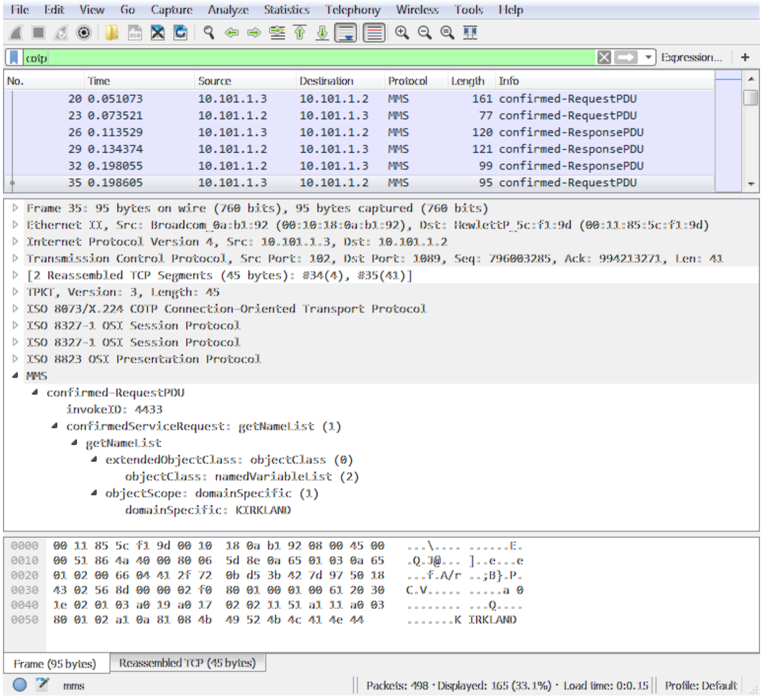
Figure 2.11: IEC 104 communication: single command type

- `Shift mode` was very similar to the `range mode`. It iterated over the range of IOAs where the new range was calculated by adding the shift values to the default range values.

- `Sequence mode` was used once they knew the values of all IOAs of the single command type. The malware immediately executed an infinite loop, sending `select and execute` packets to the IOAs defined in the config file.

61850 payload component implemented a part of IC 61850 communication standard. Similarly to the previous components, it read its configuration supplied by the `Launcher`. The config file contained a list of IP addresses of devices capable of communicating via IEC 61850 standard. If the config file was not present, the component enumerated all connected network adaptors to determine their IP subnet masks. Then it enumerated all possible IP addresses for each of these subnet masks, and tried to connect to port 102 on each of those addresses[4]. Once the component found a target, the following data communication was initiated:

1. The component sent a `Connection Request` packet using Connection Oriented Transport Protocol (COTP) [66].
2. If the target device responded correctly, it sent an `InitiateRequest` using MMS protocol [3].
3. Following that, it sent a MMS `getNameList` request to receive a list of object name in a Virtual Manufacturing Device (VMD), see Figure 2.12.



cotp

```
No.   Time        Source       Destination  Protocol  Length  Info
   20 0.051073    10.101.1.3   10.101.1.2   MMS          161  confirmed-RequestPDU
   23 0.073521    10.101.1.2   10.101.1.3   MMS           77  confirmed-RequestPDU
   26 0.113529    10.101.1.3   10.101.1.2   MMS          120  confirmed-ResponsePDU
   29 0.134374    10.101.1.2   10.101.1.3   MMS          121  confirmed-ResponsePDU
   32 0.198055    10.101.1.2   10.101.1.3   MMS           99  confirmed-ResponsePDU
   35 0.198605    10.101.1.3   10.101.1.2   MMS           95  confirmed-RequestPDU
```

```
▷ Frame 35: 95 bytes on wire (760 bits), 95 bytes captured (760 bits)
▷ Ethernet II, Src: Broadcom_0a:b1:92 (00:10:18:0a:b1:92), Dst: HewlettP_5c:f1:9d (00:11:85:5c:f1:9d)
▷ Internet Protocol Version 4, Src: 10.101.1.3, Dst: 10.101.1.2
▷ Transmission Control Protocol, Src Port: 102, Dst Port: 1089, Seq: 796003285, Ack: 994213271, Len: 41
▷ [2 Reassembled TCP Segments (45 bytes): #34(4), #35(41)]
▷ TPKT, Version: 3, Length: 45
▷ ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
▷ ISO 8327-1 OSI Session Protocol
▷ ISO 8327-1 OSI Session Protocol
▷ ISO 8823 OSI Presentation Protocol
◢ MMS
   ◢ confirmed-RequestPDU
        invokeID: 4433
      ◢ confirmedServiceRequest: getNameList (1)
         ◢ getNameList
            ◢ extendedObjectClass: objectClass (0)
                 objectClass: namedVariableList (2)
            ◢ objectScope: domainSpecific (1)
                 domainSpecific: KIRKLAND
```

```
0000  00 11 85 5c f1 9d 00 10  18 0a b1 92 08 00 45 00   ...\.... ......E.
0010  00 51 86 4a 40 00 80 06  5d 8e 0a 65 01 03 0a 65   .Q.J@... ]..e...e
0020  01 02 00 66 04 41 2f 72  0b d5 3b 42 7d 97 50 18   ...f.A/r ..;B}.P.
0030  43 02 56 8d 00 00 02 f0  80 01 00 01 00 61 20 30   C.V..... .....a 0
0040  1e 02 01 03 a0 19 a0 17  02 02 11 51 a1 11 a0 03   ........ ...Q....
0050  80 01 02 a1 0a 81 08 4b  49 52 4b 4c 41 4e 44      .......K IRKLAND
```

mms | Packets: 498 · Displayed: 165 (33.1%) · Load time: 0:0.15 | Profile: Default

Figure 2.12: Requesting `getNameList` in MMS communication

4. Then it enumerated objects and named variable in a specified domain. After that, it parsed received data searching for variables that contained string CSW which was a name for logical nodes used to control circuit breakers and switches.

[4] This is a ISO TSAP connection port for Class 0.

5. Then the component sent an additional MMS `Read` request. For some of the variables it issued a `Write` request to change its state.

Data wiper component was a destructive module that was used in the final stage of an attack to hide the tracks and make recovery difficult. This included the following activities:

- It enumerated all keys in the registry and set the value `ImagePath` with an empty string in each of the entries found to make the OS unbootable.
- It deleted files with specific extensions on all drives from `C:\` to `Z:\`. The component rewrote file content with meaningless data obtained from newly allocated memory. The files included Windows binaries (.exe, .dll), archives (.7z, .tar, .rar, .zip), backups (.bak, .bk, .bkp), MS SQL files (.mdf, .ldf), configuration files (.ini,.xml), ICS files (.scl, .cid, .scd), vendor-dependent files (SYS_BASCOM.COM for ABB), or license data (.paf).
- After deletion it terminated all processes except those included in a list of critical system processes. Then, the second attempt was made.
- Finally, the malware terminated all processes including system processes except its own. The system became unresponsive and eventually crashed.

Additional tools included *port scanner* that was used to map the network and to find computers relevant to their attack and *DoS tool* used against Siemens SIPROTEC devices. The tool sent a specially crafted UDP datagrams to port 50.000 that cause a denial-of-service of the affected device, see CVE-2015-5374[5]. Following that, the target device stopped responding to any command until it was rebooted manually.

2.4.3 Lesson Learned

Malware analysis showed that the Industroyer was an advanced and sophisticated piece of malware used against industrial control systems. It was able to directly control switches and circuit breakers at power grid substation using four ICS protocols and contained an activation timestamp for the day of the power outage. Using logs produced by the tool-set and highly configurable payload, the attackers could adapt the malware to any comparable environment. Following security recommendation were proposed by analysts [41, 17], see also Figure 2.13:

- *Visibility requirement:* Security teams should have a clear understanding of where and how IEC 104 and IEC 61850 protocols are used. They should look specifically for increased usage of the protocols against baselines established in the environment. Also, they should look for systems leveraging these protocols if they have not before and specifically try to identify systems generating new network flows using these protocols.

[5] Available at `https://nvd.nist.gov/vuln/detail/CVE-2015-5374` [August 2018]

- *Backup:* Robust backup of engineering files such as project logic, IED configuration files, and ICS application installers should be stored offline and tested. This helps reduce the impact of the wiper functionality.
- *Traffic filtering is not sufficient:* Air gapped networks, unidirectional firewalls, antivirus in the ICS, and other passive defenses and architecture changes are not appropriate solutions for this attack. No amount of security control will protect against a determined human adversary. Human defenders are required.
- *Monitoring and anomaly detection:* At key network traffic points, ensure network captures are collected, baselined, and analyzed in a manner that will identify anomalous communication. Monitor all outbound communications looking for suspicious connections. Perform network security monitoring to continuously search through the network environment for anomalies.

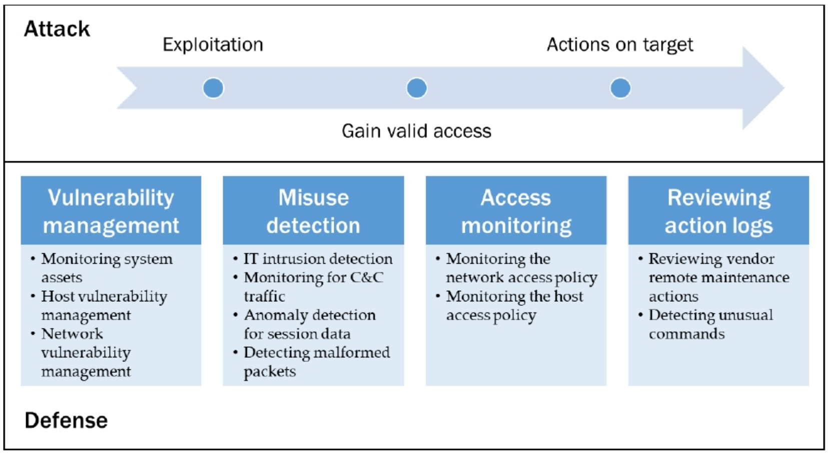

Attack
Exploitation
Actions on target
Gain valid access
Vulnerability management
• Monitoring system assets
• Host vulnerability management
• Network vulnerability management
Misuse detection
• IT intrusion detection
• Monitoring for C&C traffic
• Anomaly detection for session data
• Detecting malformed packets
Access monitoring
• Monitoring the network access policy
• Monitoring the host access policy
Reviewing action logs
• Reviewing vendor remote maintenance actions
• Detecting unusual commands
Defense

Figure 2.13: Overview of detection activities [55]

2.5 Summary

In this section we presented the architecture of typical control communication in smart grid networks and its features. We mentioned, that ICS control protocols in smart grids lack sufficient protection that would cover standard security requirements like authentication, authorization, confidentiality, data integrity, and availability. These drawbacks are addressed to a certain extent by the security standard IEC 62351 but as mentioned before, not all proposals are easy to deploy. Until this standard is fully applied to ICS protocols MMS, GOOSE, IEC 104, and others, insecure versions of these protocols will co-exist in smart grids together with those implemented security features.

We also gave an overview of recent attacks on ICS protocols. Detailed anatomy of smart grid attacks using BlackEnergy and Industroyer malware revealed that for detection and prevention of such attacks that are initiated from infected an internal device, we need to enhanced communication visibility of ICS protocols in smart grid networks together with automated detection of anomaly detection.

The following chapters will address these issues and discuss possible solution how to increase security of smart grid communication using extended ICS monitoring and application of anomaly detection on monitoring data.

Chapter 3

Increasing Visibility of ICS Communication

Network monitoring is an important part of network management. Its primary objective is to obtain monitoring data about connected devices and activities that happen on the network. By analyzing monitoring data we learn what devices are active on the network, what transmissions are established, what protocols and services are requested, what is the load of individual network devices, etc. This helps network administrators to identify potential security incidents and unusual behavior on the network.

Network monitoring of critical industrial systems like smart grid networks is mostly limited to the observation of packet or flow transmissions on Layers 3 (IP layer) and 4 (transport layer) without revealing what ICS commands are transmitted to a specific ICS device, what services have been switched on/off at a given SCADA device, or what files are downloaded from a IEC 104 master. ICS monitoring is essential for application visibility of ICS communication. ICS monitoring data represents a valuable input for intrusion and anomaly detection systems.

In this chapter we present a technique how to extend the standardized IPFIX flow monitoring system by ICS monitoring data and kinds of security incidents can be detected using these augmented IPFIX records. We introduce a definition of ICS flows and show how this definition is mapped to common ICS protocols in the smart grid, namely IEC 104, MMS, GOOSE and DLMS. Then we demonstrate on IEC 104 protocol, how ICS flow records are built and what level of details are provided by extended ICS flow monitoring. The last part of this chapter evaluates benefits of extended ICS flow monitoring for anomaly detection.

3.1 The Need for Advanced ICS Monitoring

ICS communication transmits monitoring and controlling data among industrial devices, processes and events. It was originally designed for serial data links that were physically separated from external networks. Recently, ICS protocols have been adopted to operate over Ethernet with the Internet Protocol (IP) and UDP/TCP transport. This solution opened possibility to interconnect ICS networks over wide-area networks (WANs) in order to provide remote control and online updates.

Interconnection of ICS systems with IP networks revealed serious security flaws in industrial protocol design mentioned in the previous chapter. As demonstrated on the attacks against Ukrainian power grid presented in Section 2.4.2, the malware installed on a station in the smart grid was able to masquerade as a legitimate process and communicate with RTUs without anyone noticing this. The malware scanned available resources in the substation network using address enumeration and then started to send legitimate IEC 104 commands to RTUs that manipulated circuit breakers. Since the malware was installed on a station inside the network, its communication is unnoticed by IDS systems or firewalls located at the edge of the network.

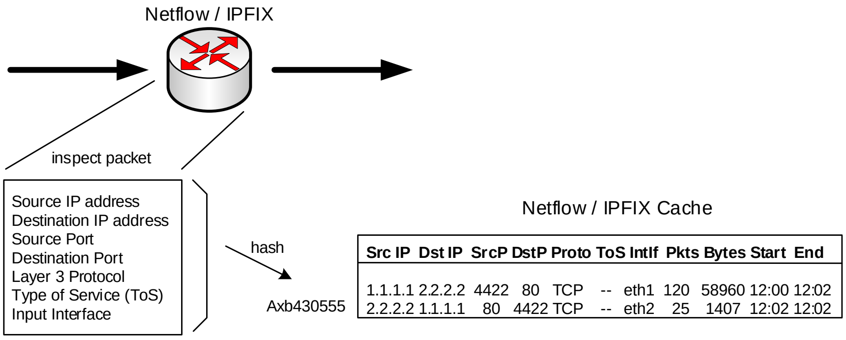
This case revealed a lack of visibility of ICS communication within the power grid [16]. Insufficiency of ICS network monitoring was also highlighted in the Report of European Union Agency for Network and Information Security (ENISA) [44]. The report warns that *without active network monitoring, it is very difficult to detect suspicious activity, identify potential threats, and quickly react to cyber attacks.* NIST Guide to ICS Security [113] recommends network segregation, application of firewall rules, redundancy, etc., to secure ICS networks. However, these techniques provides only limited security against internal cyber security attacks. For this reason, NISTIR report from 2018 focuses on behavior anomaly detection (BAD) [92].

3.1.1 IP Network Flow Monitoring

The proposed approach was inspired by IP networks where security monitoring is well established. IP monitoring techniques include SNMP monitoring [99], IP flow monitoring [31], and system logging [48]. These techniques can be applied to a certain extent also in ICS systems. Since IP monitoring relies on IP layer, it cannot be easily transferred to ICS protocols like GOOSE that run directly over link layer. Due to the restricted hardware and firmware of RTUs and IEDs, it is also not difficult to implement SNMP agents or Syslog clients on these devices and operation SNMP and Syslog monitoring in ICS networks similarly to IP networks.

A feasible option for ICS networks is IP flow monitoring using Netflow/IPFIX concept [31, 34]. This is a passive monitoring technique where a monitoring probe (exporter) observes a unidirectional sequence of packets with some common properties passing the observation point. The probe stores meta data about each flow into a flow record. Flow records include source and destination IP addresses and ports, packet and byte counts, timestamps, Type of Service (ToS), input and output interfaces, etc., see Figure 3.1. Flow records are then transported to IP flow collector where are further analyzed, visualized and processed by anomaly detection.

IPFIX standard [34] supports flexible definition of monitoring data using templates. It means that a set of observed information can be defined a user. When a



Src IP	Dst IP	SrcP	DstP	Proto	ToS	IntIf	Pkts	Bytes	Start	End
1.1.1.1	2.2.2.2	4422	80	TCP	--	eth1	120	58960	12:00	12:02
2.2.2.2	1.1.1.1	80	4422	TCP	--	eth2	25	1407	12:02	12:02

Figure 3.1: Netflow monitoring

monitoring probe implements user-defined definition of flows, it collects not only standardized values from Layer 2 to Layer 4 protocol headers, but it can also collects meta data from selected application protocols.

Application of IPFIX monitoring to ICS protocols is described in the following sections. First, we overview works related to ICS monitoring. Then we define ICS Flow and present an architecture of ICS flow monitoring system in the smart grid. The most important part of this chapter is a selection of ICS headers of smart grid protocols that creates higher visibility into ICS communication. Finally, we demonstrate, how extended ICS flow meta data increases security of smart grid communication.

3.2 Related Work

Protection of smart grid networks against cyber attacks has been researched by many authors [81, 93, 75, 91]. NIST Guide to ICS Security [113] presents a large overview of past attacks on ICS systems with recommendation how to secure ICS architecture using network segregation, firewall rules, NAT translation and other techniques.

Security of ICS/SCADA networks is often implemented by proprietary IDS systems with deep-packet inspection (DPI) that analyzes selected ICS protocols. Generally, an IDS system parses ICS packets and extracts data of interest from ICS protocol headers. The data are subject to further signature-based or behavior-based analysis. If a suspicious communication is detected, an alert is raised and the traffic is filtered out. Real-time scanning and analysis of ICS packets demand high processing power and fast memory on the analyzing device. In addition, each ICS protocol requires a specific ICS pre-processor (parser) that should be installed on an IDS system [57]. IDS systems are usually located at the perimeter of ICS/SCADA networks. This limits protection to external threats only. The proposed IPFIX flow-based monitoring system can observe communication both at the edge of the substation network as well as inside of the network.

Distributed monitoring system for protecting SCADA communication in power grid was proposed by [67]. The authors deployed SCADA probes over the power grid network which observed IEC 104 and IEC 61850 communication. The probes included Snort[1] and Bro[2] software with installed SCADA protocols analyzer. When a security incident was detected, the probe sent an alarm to the Security information and event management (SIEM) system. In contrast to their system, the proposed ICS flow monitoring can observe network communication on any place in the network. Our system also uses standardized IPFIX protocol, so it enables easy integration with current SIEM systems. Detection based on ICS-enabled IPFIX data is not limited to rule-based intrusion detection as Snort and Bro.

Barbosa, Sadre and Pras [22] proposed flow-based monitoring for whitelisting. Unlike our solution, their approach was based strictly on IP flows. They observed packets and extracted four properties to build a flow: client address, server address, server-side port and transport protocol. During the learning phase, the system created an initial white-list with legitimate flows. In the detection phase, when a new flow was detected that was previously not white-listed, an alarm was raised. Comparing to their approach, we propose monitoring of application-level data, e.g., ICS commands, objects, etc., which provides higher visibility into ICS communication.

Combination of rule-based anomaly detection and IP flow analysis is described in [75] where the authors use IP flow statistics like packet rate and packet size to classify traffic into the four behavior characteristics in power equipment. Besides, they observe GOOSE communication and detects selected security incidents using rule-based IDS. Our approach includes also application data from ICS protocols.

3.3 Flow Based Monitoring of the Smart Grid

Smart grid communication employs industrial protocols like IEC 61850 GOOSE [86], Modbus, IEC 60870-5-104 [59], DNP3, IEC 61850 MMS [3], DLMS [84] and others. These protocols transmit control and status data of industrial processes running on RTUs or IEDs. Protocols like GOOSE implement *publish-subscribe* mechanism where an application (publisher) writes the values into a local buffer that is periodically transmitted to subscribing agents using L2 multicast. ICS protocols like IEC 104, DNP3, MMS or DLMS communicate use *client-server model*. In this model, controlled station (RTU slave) is monitored or commanded by a master station. Controlling station (PC with SCADA system, RTU master) performs control of outstations. ICS client-server communication can be delivered in the *monitoring direction* (from controlled station to the controlling station) or in the *control direction* (RTU master sends commands toward the RTU slave), see Figure 3.2. This is important when analyzing ICS flows.

Smart grid security requires awareness of active communication in the network, e.g, what nodes are sending or receiving data, what ICS protocols are active in the

[1] See https://www.snort.org/ [July 2021]

[2] See https://bricata.com/blog/what-is-bro-ids/ [July 2021]

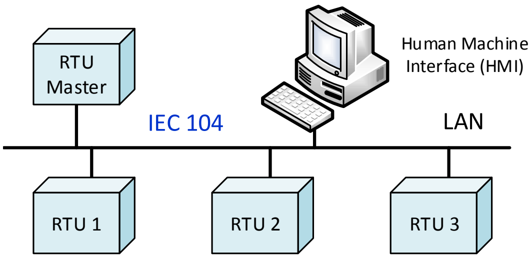


Figure 3.2: IEC 104 topology

network, what commands have been issued, how many packets were transmitted between two devices within a given time window, etc. Traditional Netflow and IPFIX monitoring provides data on Layer 2 to Layer 4. Here we show how ICS-specific data can be added to extended IPFIX records.

3.3.1 Architecture of ICS Flow Monitoring

Flow based monitoring system is composed of probes that observe packets on the link, extract meta data from passing flows and creates so called flow records that are later transmitted to the flow collector. Traditional IP flow is defined as *a sequence of IP packets passing the observation point during a certain time interval* [31]. Packets belonging to a given flow have a set of common properties. IP flow properties include source and destination IP addresses, source and destination port numbers and the protocol type. An example of the IP flow is a HTTP request from a client (identified by the source IP address and port) to a specific server (identified by the destination IP address and port). For each flow the monitoring probe collects meta data, e.g., timestamp when the first packet of the flow occurred, the number of packets in the flow, the number of transmitted bytes, duration of the flow, etc. Meta data together with flow properties are written into a *flow record*. The probe creates flow records for all flows passing the probe. The flow records are then delivered to the IPFIX collector, see Figure 3.3.

3.3.2 ICS Flow

For ICS monitoring, we extend definition of IP flows by adding property values extracted from ICS protocol headers. Let P be an IP packet with a set of IP header fields, i.e. $P = \{p_1, p_2, \ldots, p_n\}$, T be a transport layer (L4) protocol data unit (PDU) with L4 protocol headers, i.e., $T = \{t_1, t_2, \ldots, t_m\}$, and A be an application layer (L7) PDU with L7 headers, i.e., $A = \{a_1, a_2, \ldots, a_o\}$.

Definition 3.1. We define the *flow property Fprop* as a subset of selected values extracted from L3, L4 and L7 headers, i.e.,

$$Fprop(ICS) \subseteq P(IP) \cup T(UDP/TCP) \cup A(ICS) \tag{3.1}$$

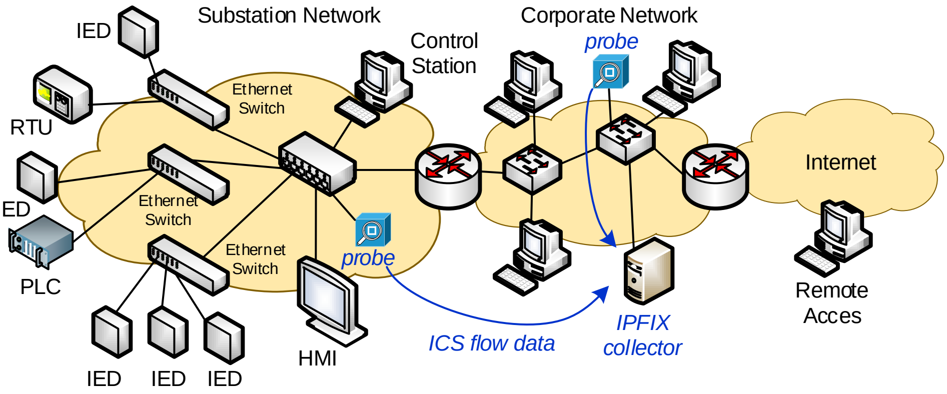


Figure 3.3: ICS flow monitoring system

The flow property *Fprop* is mapped to specific L3 and L4 that transmits ICS protocol, e.g., for IEC 104 it is the IP protocol on Layer 3 and TCP on Layer 4. Layer 7 protocol is IEC 104.

For ICS protocols transmitted directly over the link layer, e.g., GOOSE or Modbus RTU, we can either define P and T sets as empty, or we can create so-called virtual L3 and L7 layers where, for example, for L3 we derived IPv6 link local addresses from MAC addresses using EUI-64 or other techniques recommended by RFC 4291 [103].

In case of IEC 104 protocol monitoring, typical L3 properties are the source and destination IP addresses and the protocol type. Typical L4 properties include source and destination ports. On application layer, L7 properties of IEC 104 may include APDU frame type, ASDU type, cause of transmission (COT), number of information objects, origination address (ORG) and ASDU address (COA), see Figure 3.6. Thus, flow properties of IEC 104 can be expressed as follows:

$$\begin{aligned} \mathit{Fprop}(IEC104) = \{&SrcIP, DstIP, IPprot, SrcPort, DstPort,\\ &APDUtype, ASDUtype, COT, Items, ORG, COA\} \end{aligned} \tag{3.2}$$

Definition 3.1 allows flexible selection of protocol headers used for ICS flow monitoring. This is important because industrial protocols differ in protocol format and header and not all fields are equally valuable for monitoring. Thus, $Fprop$ is mapped to a given ICS protocol similarly as shown in formula (3.2).

Definition 3.2. *ICS flow* is a *sequence of ICS packets passing the observation point during a certain time and having the same flow property Fprop.*

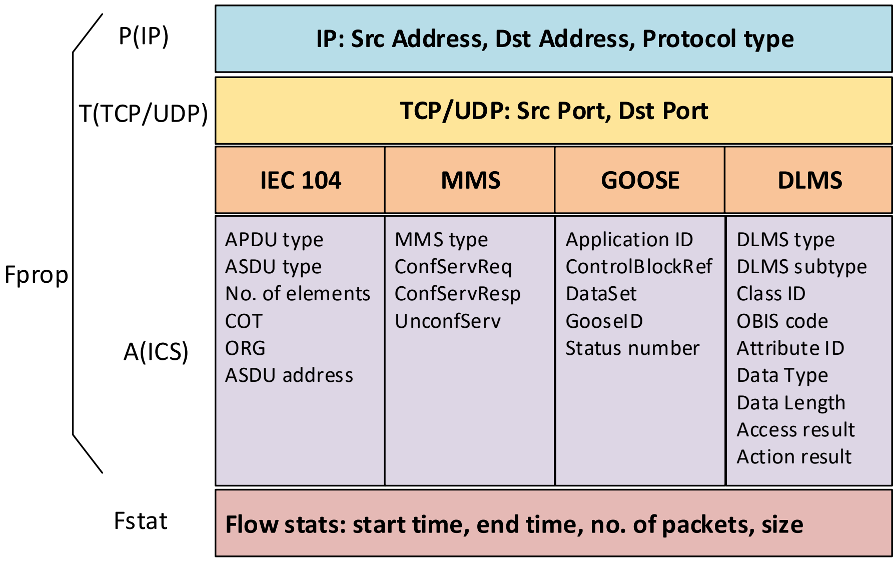
The ICS probe parses ICS packets where it extracts selected header values and insert them into the ICS flow record. The ICS flow record becomes a building block for ICS network monitoring. The ICS flow record contains $Fprop$ data that identifies the flow and statistical data $Fstat$ that describes behavior of the flow.

$Fstat$ set is computed by the probe and includes meta data like starting time of the flow, ending time, the number of packets of the flow, the total size in bytes, etc., i.e., $Fstat = \{t_{start}, t_{end}, packets, size, \ldots\}$.

Following that definitions, ICS flow record $Frec$ is a union of ICS flow property values and statistical behavior of the flow. ICS flow record is then mapped to a specific ICS protocol, e.g., IEC 104, MMS, etc.

$$Frec(ICS) = Fprop(ICS) \cup Fstat \tag{3.3}$$

The presented approach is flexible and can be applied to any ICS protocol by mapping specific protocol headers to ICS flow record fields. Table 3.1 presents recommended L7 headers of common ICS protocols that are part of smart grid communication. The headers were chosen based on protocol behavior and application domain knowledge.



Fprop	P(IP)	IP: Src Address, Dst Address, Protocol type			
	T(TCP/UDP)	TCP/UDP: Src Port, Dst Port			
		IEC 104	MMS	GOOSE	DLMS
	A(ICS)	APDU type ASDU type No. of elements COT ORG ASDU address	MMS type ConfServReq ConfServResp UnconfServ	Application ID ControlBlockRef DataSet GooseID Status number	DLMS type DLMS subtype Class ID OBIS code Attribute ID Data Type Data Length Access result Action result
Fstat		Flow stats: start time, end time, no. of packets, size			

Table 3.1: Recommended ICS headers extracted from ICS protocols.

As mentioned above, ICS monitoring requires implementation of a ICS protocol parser for each supported protocol and definition of an IPFIX template that maps $Frec(ICS)$ fields into IPFIX record format. Parsers for common smart grid protocols GOOSE, IEC-104, MMS and DLMS were implemented in frame of the research project IRONSTONE (2016-2019) and are available at the project web site[3]. An example of the IEC 104 template is in Appendix B.1. The next part will thoroughly demonstrate how ICS flows are created for IEC 104 protocol.

[3] See https://www.fit.vut.cz/research/project/1101/.en [Dec 2019]

3.4 Example: ICS Flows for IEC 104 Protocol

IEC 104 protocol is a part of IEC Telecontrol Equipment and Systems standard IEC 60870-5 that provides a communication profile for sending basic telecontrol messages between two systems in electrical engineering and power system automation [59]. IEC 104 operates over TCP using client-server communication model and delivers supervisory data and acquisition request for controlling power grids.

IEC 104 messages are exchanged between the controlled and the controlling station. *Controlled station* (also called outstation, RTU slave) is monitored or commanded by a master station. *Controlling station* (typically a PC with SCADA system, RTU master) performs control of outstations. IEC 104 communication is delivered in the *monitoring direction*, i.e., from controlled station to the controlling station, or in the *control direction*.

3.4.1 IEC 104 Protocol

IEC 104 protocol [59] is implemented on application layer of TCP/IP stack using Application Protocol Data Unit (APDU) and Application Service Data Unit (ASDU), see Figure 3.4 and Appendix A.1. Based on APDU Control Field 1, three APDU formats are defined: *I-frames* for transmitting data, *S-frames* for numbered supervisory operations, and *U-frames* transmitting unnumbered control functions (test frame, start transfer, stop transfer).

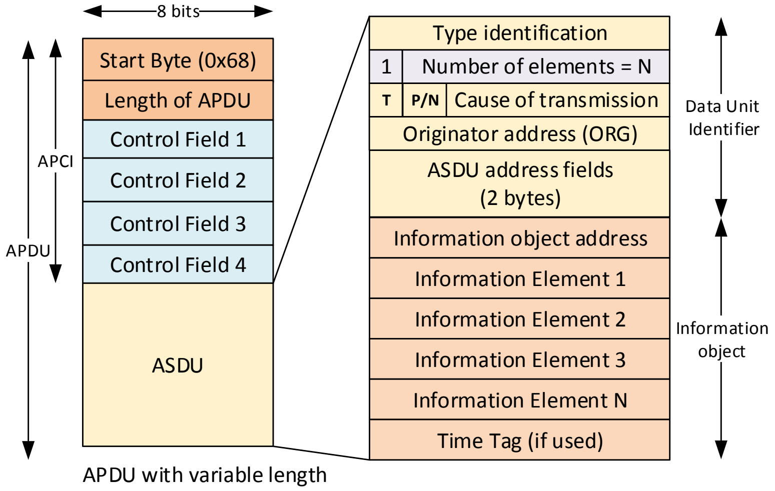
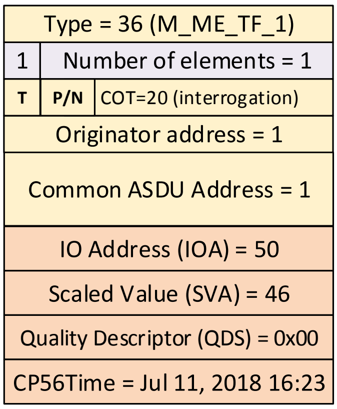


Figure 3.4: IEC 104 protocol header

The most important APDUs for security monitoring are I-frames that transmit ASDUs. The ASDU includes two sections: fixed-length ASDU header and a variable-length list of information objects. The header includes *ASDU type* (e.g., single point of information, measured value, regulating step command, read command), *number of transmitted objects*, *cause of transmission* (COT, e.g., periodic, spontaneous, activation, interrogation) and *ASDU address* (station address).

Each *information object* is addressed by *Information Object Address* (IOA) that identifies particular data on the given node. For each ASDU type, the IEC 104 standard defines the format of information object, that includes information elements which form the object and structure of the data. For example, information object in ASDU with type 36 (Measured value, scaled value with time tag) and cause of transmission 20 (interrogation) contains two information elements: SVA (Scaled value) and QDS (Quality descriptor), see Figure 3.5.



Type = 36 (M_ME_TF_1)

1 | Number of elements = 1

T | P/N | COT=20 (interrogation)

Originator address = 1

Common ASDU Address = 1

IO Address (IOA) = 50

Scaled Value (SVA) = 46

Quality Descriptor (QDS) = 0x00

CP56Time = Jul 11, 2018 16:23

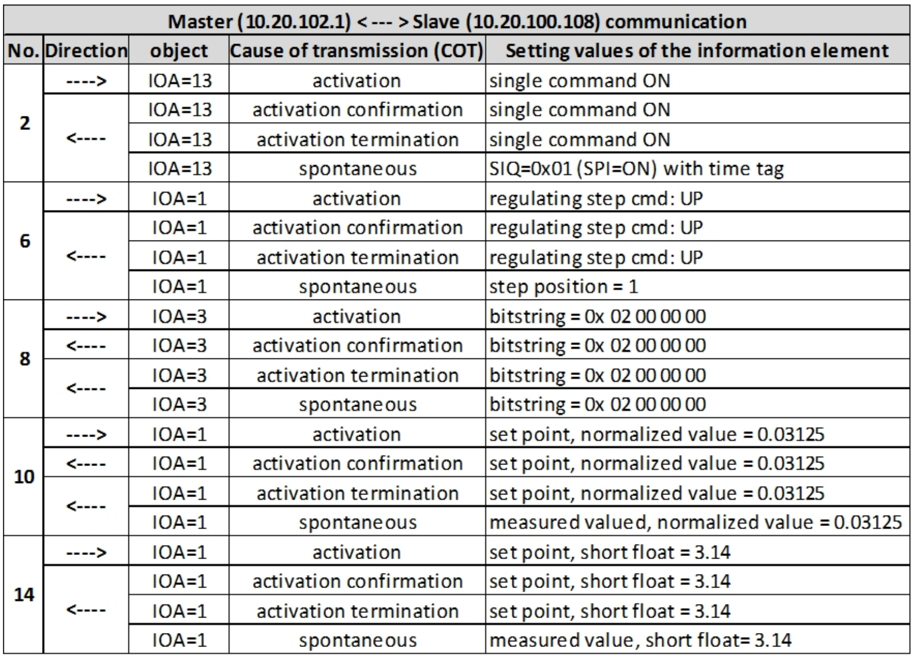
Figure 3.5: Example of IEC 104 packet

Typical IEC communication is depicted in Table 3.2 that shows selected IEC 104 transactions (transactions no. 2,6,8,10,14) exchanged between the IEC 104 master and slave in both directions: control and monitoring. Each transaction contains at least one ASDU with an object addressed by the Information Object Address (IOA), Cause of Transmission (COT) and transmitted value or command. You can notice that one ASDU may transmit several IOA objects with different IOA addresses, COTs and data. The proposed monitoring system retrieves interesting data from IEC 104 packets in order to create IEC 104 flows for ICS monitoring.

3.4.2 Defining IEC 104 Flow Records

As previously stated, the traffic flow includes Layer 3 and Layer 4 data only, see [32, 34]. For increasing IEC 104 visibility, we add the following IEC 104 headers to the IEC 104 flow definition:

- APDU frame type
- ASDU type
- ASDU cause of transmission (COT)
- Number of information object
- Originator address (ORG)
- ASDU address (COA)



Master (10.20.102.1) < --- > Slave (10.20.100.108) communication				
No.	Direction	object	Cause of transmission (COT)	Setting values of the information element
2	---->	IOA=13	activation	single command ON
	<----	IOA=13	activation confirmation	single command ON
		IOA=13	activation termination	single command ON
		IOA=13	spontaneous	SIQ=0x01 (SPI=ON) with time tag
6	---->	IOA=1	activation	regulating step cmd: UP
	<----	IOA=1	activation confirmation	regulating step cmd: UP
		IOA=1	activation termination	regulating step cmd: UP
		IOA=1	spontaneous	step position = 1
8	---->	IOA=3	activation	bitstring = 0x 02 00 00 00
	<----	IOA=3	activation confirmation	bitstring = 0x 02 00 00 00
	<----	IOA=3	activation termination	bitstring = 0x 02 00 00 00
		IOA=3	spontaneous	bitstring = 0x 02 00 00 00
10	---->	IOA=1	activation	set point, normalized value = 0.03125
	<----	IOA=1	activation confirmation	set point, normalized value = 0.03125
	<----	IOA=1	activation termination	set point, normalized value = 0.03125
		IOA=1	spontaneous	measured valued, normalized value = 0.03125
14	---->	IOA=1	activation	set point, short float = 3.14
	<----	IOA=1	activation confirmation	set point, short float = 3.14
		IOA=1	activation termination	set point, short float = 3.14
		IOA=1	spontaneous	measured value, short float= 3.14

Table 3.2: Example of IEC 104 communication

3.4.3 Virtual Flows

IP flow monitoring works with IP packet. For each flow, it observes all packets belonging to the flow and creates one record for that flow. IEC 104 communication, however, may encapsulate several ASDUs into one TCP packet. Using the standard definition of IP flow, we should create only one flow record for this packet, i.e., only data from the first ASDU, see Section 3.4.2, would be included in the flow and other ASDUs ignored, or all ASDUs should be aggregated into on record. In both case we would loose detailed information about transmitted ASDUs.

To deal with this issue, we need to split an IP flow with multiple ASDUs into multiple IEC 104 virtual flows where each IEC 104 virtual flow transmits exactly one ASDU. On one side, this will increase the number of monitoring flows, on the other side, the required IEC 104 visibility will be preserved.

3.4.4 Building IEC 104 Flows

Figure 3.6 shows how IEC 104 flow records are created from IEC 104 communication. The flow is composed of five packets. The IEC 104 flow record contains values from Layer 3 (IP addresses), Layer 4 (ports) and Layer 7 (selected IEC 104 headers). All these property values identify an IEC 104 flow. The flow record also includes statistical values related to the flow (timestamps, size, bytes, etc.). After the final packet of IEC 104 flow is observed by the monitoring probe, the flow is closed and transmitted by IPFIX protocol to the IPFIX collector. A set of

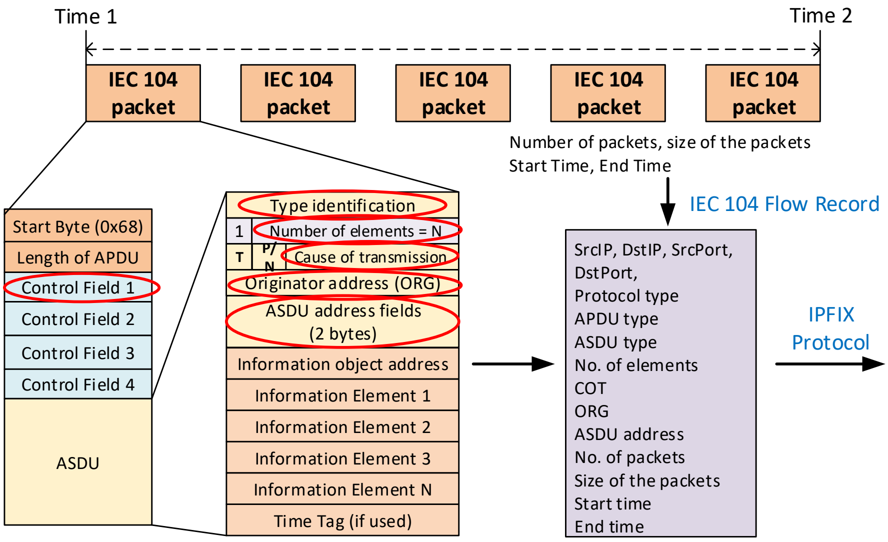


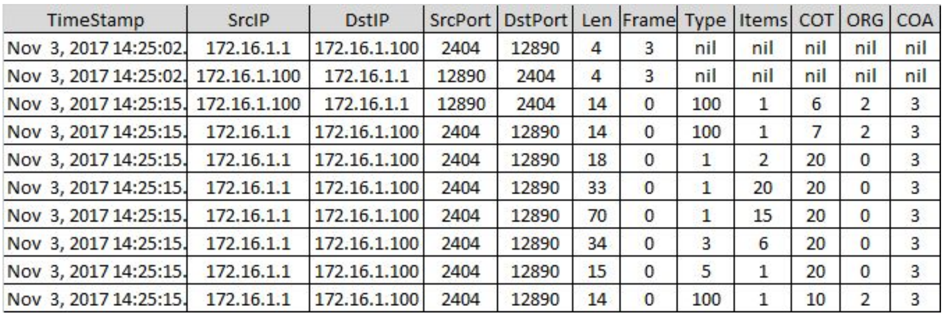
Figure 3.6: Building an IEC 104 flow record from IEC 104 packets.

flow records is then analyzed and visualized through network management system. Technical details about creating IEC 104 flows are mentioned at [88].

3.4.5 Collecting ICS Flow Data

A big advantage of ICS flow-based monitoring is that ICS monitoring probes can be deployed anywhere in the smart grid network as shown in Figure 3.3, thus providing monitoring data from overall the network.

Table 3.3 shows a raw format of IEC 104 flows extracted from IEC 104 communication. For space limit, not all flow items are displayed. Besides IP addresses and ports, IEC 104 flow record includes APDU length (Len), APDU frame format (Frame), ASDU type (Type), the number of Information Objects (Items), the Cause of Transmission (COT), Originator Address (ORG), and the ASDU address (COA).



TimeStamp	SrcIP	DstIP	SrcPort	DstPort	Len	Frame	Type	Items	COT	ORG	COA
Nov 3, 2017 14:25:02.	172.16.1.1	172.16.1.100	2404	12890	4	3	nil	nil	nil	nil	nil
Nov 3, 2017 14:25:02.	172.16.1.100	172.16.1.1	12890	2404	4	3	nil	nil	nil	nil	nil
Nov 3, 2017 14:25:15.	172.16.1.100	172.16.1.1	12890	2404	14	0	100	1	6	2	3
Nov 3, 2017 14:25:15.	172.16.1.1	172.16.1.100	2404	12890	14	0	100	1	7	2	3
Nov 3, 2017 14:25:15.	172.16.1.1	172.16.1.100	2404	12890	18	0	1	2	20	0	3
Nov 3, 2017 14:25:15.	172.16.1.1	172.16.1.100	2404	12890	33	0	1	20	20	0	3
Nov 3, 2017 14:25:15.	172.16.1.1	172.16.1.100	2404	12890	70	0	1	15	20	0	3
Nov 3, 2017 14:25:15.	172.16.1.1	172.16.1.100	2404	12890	34	0	3	6	20	0	3
Nov 3, 2017 14:25:15.	172.16.1.1	172.16.1.100	2404	12890	15	0	5	1	20	0	3
Nov 3, 2017 14:25:15.	172.16.1.1	172.16.1.100	2404	12890	14	0	100	1	10	2	3

Table 3.3: Example IEC 104 flows (selected fields only)

Fields with nil value in the first two rows indicate IEC 104 APDUs without the ASDU payload, i.e., U-frames (frame type=3). Only I-frames (frame type=0) transmit ASDUs as showed in Figure 3.6.

From the list of IEC 104 flows above, we can notice that there is a controlling station with originator address ORG=0 running on IP address 172.16.1.100. The third flow record describes an I-frame (Frame=0) that encapsulates ASDU data sent by a controlling station with ORG=2 to the controlled station with ASDU address 3. Type of this ASDU is 100 (interrogation command) and cause of transmission (COT) is 6 (Activation). The controlled station responses with COT=7 (Activation Confirmation). Then we see packets transmitted in monitoring direction from station 172.16.1.1 with originator address ORG=0 to station 172.16.1.100. TypeID=1 means `Single Point Information`, typeID=3 `Double Point Information`, and typeID=5 `Step Position Information`. The station sends monitoring data of active information objects with the Cause of Transmission COT=20 (Interrogation). Value `Items` gives a number of Information Objects transmitted within the ASDU.

ICS flow records contain a list of active stations and commands that were exchanged which enhances visibility in of ICS communication in the smart grid.

3.5 Comparison of IP Flows and ICS Flows

In this section we show benefits of ICS flows monitoring approach with respect to the traditional IP flow monitoring. The section discusses various levels of details that can be obtained from ICS protocol headers. Naturally, with the increased number of details, more processing power and time is required at the monitoring probe. On the other hand, by reducing the number of observed ICS headers, we will lost communication details. The following part discusses how various levels of details in ICS flow records impact the visibility of ICS communication.

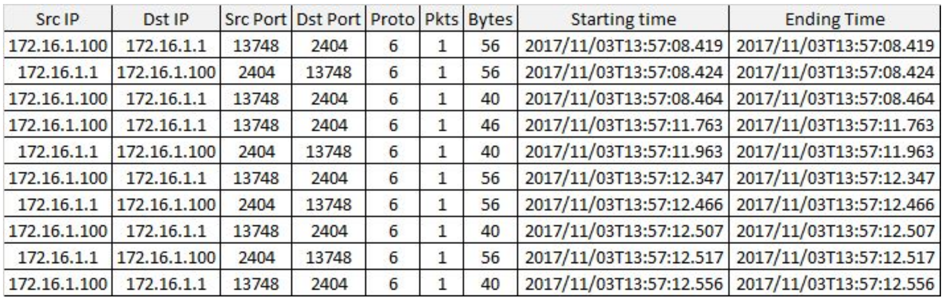
The case will be demonstrated on IEC 104 traffic that was captured during the cyber attack against the IED device. The attacker sent multiple IEC 104 activation commands in order to switch the device off. The attack is expressed by sending repeated sequence of ASDUs with the Cause of Transmission 6 (Activation), 7 (Activation Confirmation), and 10 (Activation Termination) and with the `Double Command` ASDU that invoked a switch on/off function on the target device.

3.5.1 IP flows

As mentioned before, traditional IP flow monitoring processes only Layer 3 and Layer 4 headers, e.g., IP addresses and ports. Table 3.4 shows IP flows created during the attack using the Silk Netflow/IPFIX probe[4].

The Silk creates a unique IP flow for each ASDU packet without parsing application data. When monitoring the attack, we may notice intensive activity on the

[4] See `https://tools.netsa.cert.org/silk/` [May 2019].

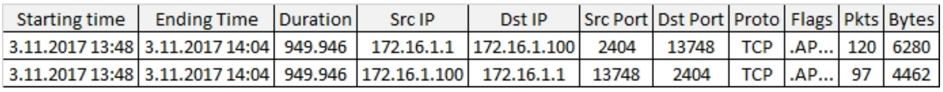


Src IP	Dst IP	Src Port	Dst Port	Proto	Pkts	Bytes	Starting time	Ending Time
172.16.1.100	172.16.1.1	13748	2404	6	1	56	2017/11/03T13:57:08.419	2017/11/03T13:57:08.419
172.16.1.1	172.16.1.100	2404	13748	6	1	56	2017/11/03T13:57:08.424	2017/11/03T13:57:08.424
172.16.1.100	172.16.1.1	13748	2404	6	1	40	2017/11/03T13:57:08.464	2017/11/03T13:57:08.464
172.16.1.100	172.16.1.1	13748	2404	6	1	46	2017/11/03T13:57:11.763	2017/11/03T13:57:11.763
172.16.1.1	172.16.1.100	2404	13748	6	1	40	2017/11/03T13:57:11.963	2017/11/03T13:57:11.963
172.16.1.100	172.16.1.1	13748	2404	6	1	56	2017/11/03T13:57:12.347	2017/11/03T13:57:12.347
172.16.1.1	172.16.1.100	2404	13748	6	1	56	2017/11/03T13:57:12.466	2017/11/03T13:57:12.466
172.16.1.100	172.16.1.1	13748	2404	6	1	40	2017/11/03T13:57:12.507	2017/11/03T13:57:12.507
172.16.1.1	172.16.1.100	2404	13748	6	1	56	2017/11/03T13:57:12.517	2017/11/03T13:57:12.517
172.16.1.100	172.16.1.1	13748	2404	6	1	40	2017/11/03T13:57:12.556	2017/11/03T13:57:12.556

Table 3.4: IP flows during the attack obtained by Silk.

link, however, we cannot determine the cause of this activity and what effect it has on ICS devices.

Creating flows for each single ASDU is not typical IP flow behavior, see Section 3.4.3. The standard IP flow includes five key properties (srcIP, dstIP, srcPort, dstPort, Protocol). Thus, the attack communication would yield two IP flows only, see Table 3.5. The result was obtained using the softflowd probe[5].



Starting time	Ending Time	Duration	Src IP	Dst IP	Src Port	Dst Port	Proto	Flags	Pkts	Bytes
3.11.2017 13:48	3.11.2017 14:04	949.946	172.16.1.1	172.16.1.100	2404	13748	TCP	.AP...	120	6280
3.11.2017 13:48	3.11.2017 14:04	949.946	172.16.1.100	172.16.1.1	13748	2404	TCP	.AP...	97	4462

Table 3.5: IP flows during the attack obtained by softflowd.

This example demonstrates limits of traditional IP flow monitoring which is not able to provide higher visibility of ICS communication and reveal ICS-specific attacks.

3.5.2 ICS flows with standard ICS headers

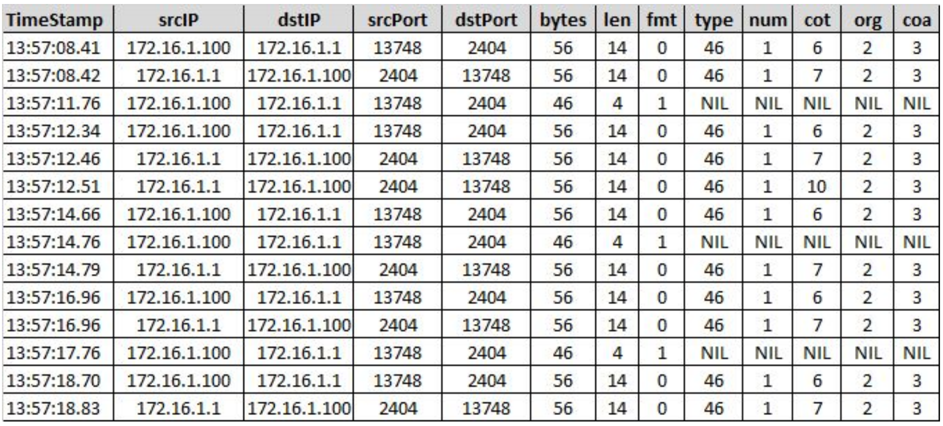
ICS flows monitoring extends monitoring data with selected L7 headers as described in Section 3.3.2. By analyzing ICS flows on the traffic with an attack, the attacker's activity are revealed, see Table 3.6.

The ICS flow records shows that a sending node with IP 172.16.1.100 sends `Double Command` operation (type=46) to the IEC device with address 3 (COA). This level of details does not provide information which object on the device was requested, however, they present valuable source of information for statistical-based anomaly detection and behavior-based anomaly detection as described later.

3.5.3 ICS flows with extended headers

The ICS-enabled probe can also implement advanced ICS protocol pre-processing that extracts additional ICS headers from the packets. These new headers extend

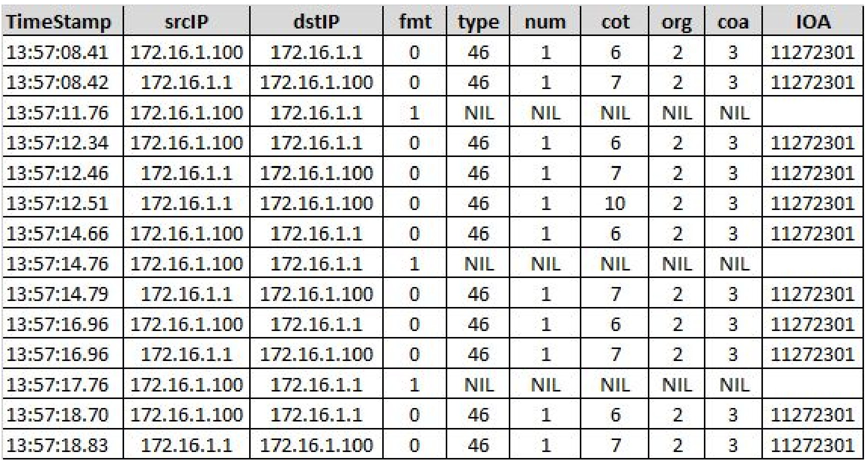
[5] See https://github.com/irino/softflowd [May 2019].



TimeStamp	srcIP	dstIP	srcPort	dstPort	bytes	len	fmt	type	num	cot	org	coa
13:57:08.41	172.16.1.100	172.16.1.1	13748	2404	56	14	0	46	1	6	2	3
13:57:08.42	172.16.1.1	172.16.1.100	2404	13748	56	14	0	46	1	7	2	3
13:57:11.76	172.16.1.100	172.16.1.1	13748	2404	46	4	1	NIL	NIL	NIL	NIL	NIL
13:57:12.34	172.16.1.100	172.16.1.1	13748	2404	56	14	0	46	1	6	2	3
13:57:12.46	172.16.1.1	172.16.1.100	2404	13748	56	14	0	46	1	7	2	3
13:57:12.51	172.16.1.1	172.16.1.100	2404	13748	56	14	0	46	1	10	2	3
13:57:14.66	172.16.1.100	172.16.1.1	13748	2404	56	14	0	46	1	6	2	3
13:57:14.76	172.16.1.100	172.16.1.1	13748	2404	46	4	1	NIL	NIL	NIL	NIL	NIL
13:57:14.79	172.16.1.1	172.16.1.100	2404	13748	56	14	0	46	1	7	2	3
13:57:16.96	172.16.1.100	172.16.1.1	13748	2404	56	14	0	46	1	6	2	3
13:57:16.96	172.16.1.1	172.16.1.100	2404	13748	56	14	0	46	1	7	2	3
13:57:17.76	172.16.1.100	172.16.1.1	13748	2404	46	4	1	NIL	NIL	NIL	NIL	NIL
13:57:18.70	172.16.1.100	172.16.1.1	13748	2404	56	14	0	46	1	6	2	3
13:57:18.83	172.16.1.1	172.16.1.100	2404	13748	56	14	0	46	1	7	2	3

Table 3.6: IEC 104 flows during the attack.

a set of ICS flow record values $Fprop$. In this case, the probe extracts the IOA address of an information object that is involved in communication, see Table 3.7.

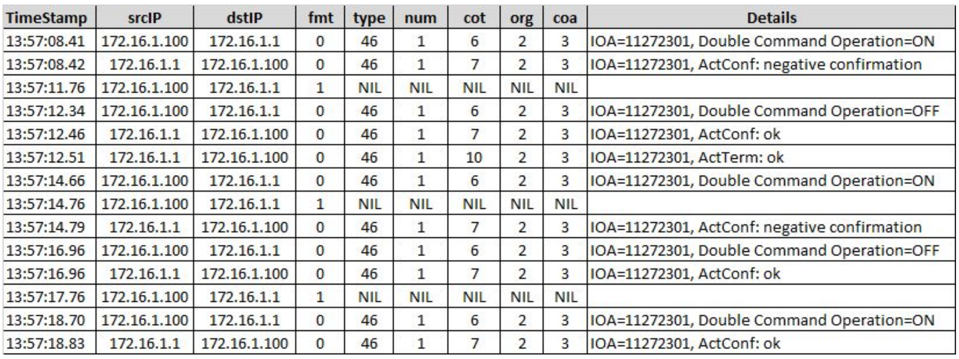


TimeStamp	srcIP	dstIP	fmt	type	num	cot	org	coa	IOA
13:57:08.41	172.16.1.100	172.16.1.1	0	46	1	6	2	3	11272301
13:57:08.42	172.16.1.1	172.16.1.100	0	46	1	7	2	3	11272301
13:57:11.76	172.16.1.100	172.16.1.1	1	NIL	NIL	NIL	NIL	NIL	
13:57:12.34	172.16.1.100	172.16.1.1	0	46	1	6	2	3	11272301
13:57:12.46	172.16.1.1	172.16.1.100	0	46	1	7	2	3	11272301
13:57:12.51	172.16.1.1	172.16.1.100	0	46	1	10	2	3	11272301
13:57:14.66	172.16.1.100	172.16.1.1	0	46	1	6	2	3	11272301
13:57:14.76	172.16.1.100	172.16.1.1	1	NIL	NIL	NIL	NIL	NIL	
13:57:14.79	172.16.1.1	172.16.1.100	0	46	1	7	2	3	11272301
13:57:16.96	172.16.1.100	172.16.1.1	0	46	1	6	2	3	11272301
13:57:16.96	172.16.1.1	172.16.1.100	0	46	1	7	2	3	11272301
13:57:17.76	172.16.1.100	172.16.1.1	1	NIL	NIL	NIL	NIL	NIL	
13:57:18.70	172.16.1.100	172.16.1.1	0	46	1	6	2	3	11272301
13:57:18.83	172.16.1.1	172.16.1.100	0	46	1	7	2	3	11272301

Table 3.7: IEC 104 flows extended by IOA address.

Pre-processing can go further. Table 3.8 shows IEC 104 flow records with deeper analysis of information elements transmitted in ASDUs.

As stated before, more detailed pre-processing of packet headers requires higher computational power on the probe. In addition, more items in IPFIX template of the ICS flow increase the size of transmitted IPFIX records and require more storage space at the collector. Thus, it is necessary to find a balance between the level of details required for ICS monitoring, and hardware capability.



TimeStamp	srcIP	dstIP	fmt	type	num	cot	org	coa	Details
13:57:08.41	172.16.1.100	172.16.1.1	0	46	1	6	2	3	IOA=11272301, Double Command Operation=ON
13:57:08.42	172.16.1.1	172.16.1.100	0	46	1	7	2	3	IOA=11272301, ActConf: negative confirmation
13:57:11.76	172.16.1.100	172.16.1.1	1	NIL	NIL	NIL	NIL	NIL	
13:57:12.34	172.16.1.100	172.16.1.1	0	46	1	6	2	3	IOA=11272301, Double Command Operation=OFF
13:57:12.46	172.16.1.1	172.16.1.100	0	46	1	7	2	3	IOA=11272301, ActConf: ok
13:57:12.51	172.16.1.1	172.16.1.100	0	46	1	10	2	3	IOA=11272301, ActTerm: ok
13:57:14.66	172.16.1.100	172.16.1.1	0	46	1	6	2	3	IOA=11272301, Double Command Operation=ON
13:57:14.76	172.16.1.100	172.16.1.1	1	NIL	NIL	NIL	NIL	NIL	
13:57:14.79	172.16.1.1	172.16.1.100	0	46	1	7	2	3	IOA=11272301, ActConf: negative confirmation
13:57:16.96	172.16.1.100	172.16.1.1	0	46	1	6	2	3	IOA=11272301, Double Command Operation=OFF
13:57:16.96	172.16.1.1	172.16.1.100	0	46	1	7	2	3	IOA=11272301, ActConf: ok
13:57:17.76	172.16.1.100	172.16.1.1	1	NIL	NIL	NIL	NIL	NIL	
13:57:18.70	172.16.1.100	172.16.1.1	0	46	1	6	2	3	IOA=11272301, Double Command Operation=ON
13:57:18.83	172.16.1.1	172.16.1.100	0	46	1	7	2	3	IOA=11272301, ActConf: ok

Table 3.8: IEC 104 flows with information elements.

3.6 Visibility of Smart Grid Communication

The level of visibility of ICS communication depends on implementation of a ICS pre-processor and operational requirements. Figure 3.7 displays multiple levels of ICS flow monitoring applied to IEC 104 communication.

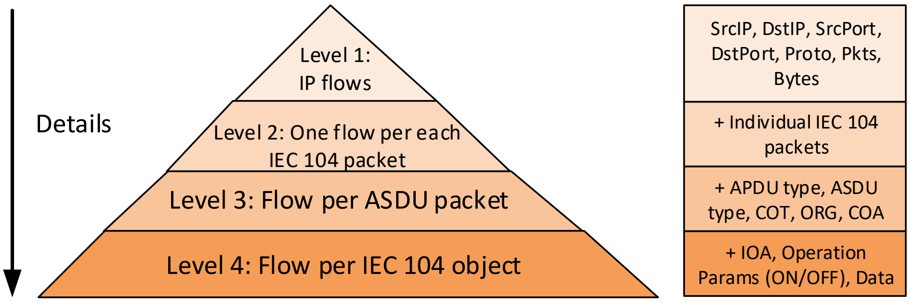

Details
Level 1: IP flows
Level 2: One flow per each IEC 104 packet
Level 3: Flow per ASDU packet
Level 4: Flow per IEC 104 object
SrcIP, DstIP, SrcPort, DstPort, Proto, Pkts, Bytes
+ Individual IEC 104 packets
+ APDU type, ASDU type, COT, ORG, COA
+ IOA, Operation Params (ON/OFF), Data

Figure 3.7: Levels of IEC 104 visibility.

As you can see, traditional IP flow monitoring offers only basic statistics about communication between two hosts extracted from Layer 3 and 4 PDUs. This level of monitoring does not provide sufficient visibility of ICS transmissions, as demonstrated on attack scenario in Section 3.5.1. Nevertheless, traditional IP flows provide useful data that may reveal a few types of common cyber attacks, e.g., detection of a rogue device on the network, DoS attack, network scanning, etc. However, without L7 information, we are not able to disclose details of the attack and identify possible consequences..

By splitting the IP flow into ICS protocol-based flows as implemented by Silk, see Table 3.4, we receive detailed statistics about individual L7 packets without knowing what operation was requested, what IOA objects were involved in communication, etc. These flow records include Layer 3 and 4 headers only as in the previous case. No real ICS visibility is provided on that level.

The Level 3 of ICS monitoring requires processing of ICS headers by the monitoring probe. Extracted ICS headers are then included into ICS flow records. In case of IEC 104 communication, it means adding APDU type, ASDU type, Cause of Transmission, Originator Address, and Common Address of ASDU (COA). This level of visibility is sufficient for observing details about daily transmission in the IEC 104 network.

Using Level 3 ICS monitoring data, we can monitor regular behavior of communicating nodes, detect if IEC 104 commands are properly transmitted in control or monitoring direction, check the names of requested information objects, detect IEC 104 resource scanning and provide enriched input data for statistical-based or behavior-based anomaly detection.

The most detailed level of ICS visibility is obtained by processing all embedded objects in the ICS protocol. In case of IEC 104 communication, Level 4 monitoring provides data about information objects and information elements transmitted in ASDU packets. Such ICS packet processing almost corresponds to the full packet capturing that processes both the ICS packet header and the payload.

Full packet processing requires high CPU performance and big memory if deployed on high-speed links. In case of links with limited bandwidth (around 100 Mb/s) a software-based monitoring probe is sufficient. Application level processing on high-speed networks (tens of Gb/s) requires accelerated hardware using FPGA or ASIC. Nevertheless, ICS communication does not require higher speed links, so even Level 4 packet processing can be implemented in software. Our experiments also demonstrate that Level 3 provides sufficient visibility of ICS communication for most use cases.

3.7 Security Monitoring Using ICS Flow Records

This section demonstrates how common security incidents in smart grid networks can be detected using ICS flow records. Instead of creating a comprehensive threat model that considers all aspects related to these attacks we focus on threat categories listed in the NISTIR 8219 report [92].

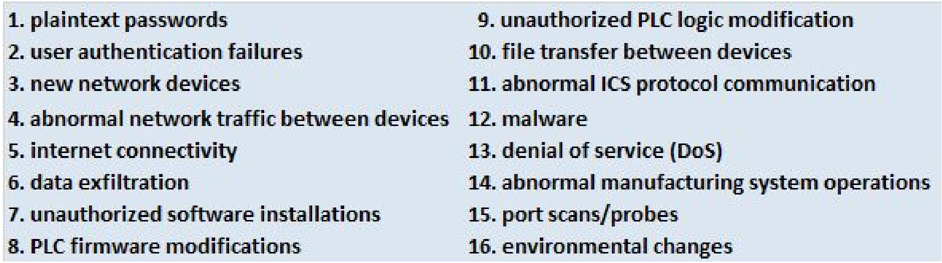
The report aims to evaluation of available techniques for the identification of activities that can be a part of attack scheme. The listed activities thus represent a representative sample of operations used by different types of attackers in the course of an attack. Our goal is to demonstrate that flow-based security monitoring with simple anomaly detection is able to identify such activities.

The detection will be demonstrated on IEC 104 datasets using simple statistical techniques. However, the presented approach is flexible and can be applied to any ICS protocol by mapping specific protocol headers to ICS flow records as proposed in Section 3.3.2.

3.7.1 Detecting Security Incidents

The NIST report *Securing Manufacturing Industrial Control Systems: Behavioral Anomaly Detection* [92] presents practical approaches for strengthening cyber security in the manufacturing processes using behavioral anomaly detection (BAD). Similar to our approach, the NIST report presents non-intrusive techniques to analyze industrial network communications. Passive monitoring can be implemented via port mirroring. The same solution uses the ICS flow monitoring probe that passively observes the traffic mirrored to the probe.

We argue that ICS flows are sufficient to cover majority of BAD classes described in the NISTIR report. The list of BAD capabilities observed by NIST is summarized in Table 3.9.



1. plaintext passwords
2. user authentication failures
3. new network devices
4. abnormal network traffic between devices
5. internet connectivity
6. data exfiltration
7. unauthorized software installations
8. PLC firmware modifications
9. unauthorized PLC logic modification
10. file transfer between devices
11. abnormal ICS protocol communication
12. malware
13. denial of service (DoS)
14. abnormal manufacturing system operations
15. port scans/probes
16. environmental changes

Table 3.9: Behavioral Anomaly Detection (BAD) classes [92].

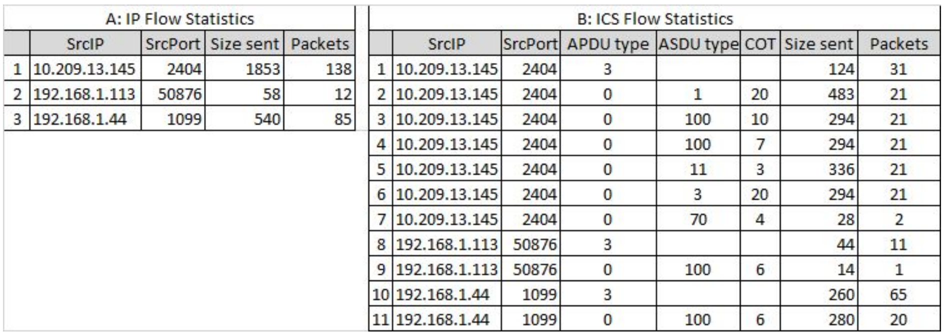
The NIST report documents the use of BAD capabilities in two environments: a robotics-based manufacturing system and process control system in chemical industry. We apply the selected scenarios of the report on IEC 104 communication.

3.7.1.1 Rogue Device Detection

The ICS flow data provides sufficient visibility that enables to detect rogue devices on the network. Generally, ICS networks show signs of stability in the number of connected devices as observed by [18]. Using flow based monitoring, active ICS nodes can be learned from TCP handshake [22]. This helps to determine which station is a client, which is a server and what type of communication is established. The authors applied whitelisting on flow data that compared newly detected devices with the list of known devices. This approach worked well for ICS protocols over TCP but it could not be applied to L2 protocols like GOOSE or Modbus RTU.

The proposed model of ICS flow monitoring creates ICS flows even for ICS protocols directly encapsulated in Ethernet like GOOSE or transmitted over serial links like Modbus RTU or IEC 101, see Section 3.3.2. By analyzing transmitted additional ICS data like ASDU type, we can check if the packet was transmitted in monitor direction (from the RTU slave to the RTU master) or in control direction (from the RTU master to the RTU slave). Thus, we do not rely on TCP data only, e.g, packet type, registered port number, but we check if the command was transmitted in the right direction.

Table 3.10 shows how rogue devices are identified using ICS flow data. Part A represents IP flows aggregated by source IP address and port. We can see three communicating nodes without being able to identify their roles. Part B shows ICS flows. It reveals communication details about the APDU type, ASDU type, and the Cause of Transmission (COT). Flows with ASDU type 1 (`Single Point of Information`), 2 (`Double Point of Information`, 11 (`Measured Values`), or 70 (`End of Initialization`) are usually initiated by the RTU slave.



A: IP Flow Statistics

	SrcIP	SrcPort	Size sent	Packets
1	10.209.13.145	2404	1853	138
2	192.168.1.113	50876	58	12
3	192.168.1.44	1099	540	85

B: ICS Flow Statistics

	SrcIP	SrcPort	APDU type	ASDU type	COT	Size sent	Packets
1	10.209.13.145	2404	3			124	31
2	10.209.13.145	2404	0	1	20	483	21
3	10.209.13.145	2404	0	100	10	294	21
4	10.209.13.145	2404	0	100	7	294	21
5	10.209.13.145	2404	0	11	3	336	21
6	10.209.13.145	2404	0	3	20	294	21
7	10.209.13.145	2404	0	70	4	28	2
8	192.168.1.113	50876	3			44	11
9	192.168.1.113	50876	0	100	6	14	1
10	192.168.1.44	1099	3			260	65
11	192.168.1.44	1099	0	100	6	280	20

Table 3.10: Analyzing ASDU types of IEC 104 communication

From flow records in Table 3.10 (B) we can see flows with ASDU type 100 (`Interrogation Command`) that ares sent from IP address 10.209.13.145 and port 2404 in control direction. Generally, using ICS flow records, we can verify the role of the device. For example, we can identify the RTU master not only by observing the reserved port 2404 used for RTU masters, but also by a command that was sent. IEC 104 defines what commands are sent in control direction (from the master) and what commands are sent in monitor direction (from the slave).

This means, that the anomaly detection system in the smart grid learns IP addresses and roles of all communicating nodes connected to the network during the learning phase. Then, when an unknown device or a device with unexpected role is detected, an alarm would be raised with details about the intruder.

3.7.1.2 Abnormal Network Traffic

ICS traffic exhibits long-term stability and periodicity as observed by [122, 19, 20] and can be described using communication patterns [78]. Any traffic with unusual behavior, e.g., an invalid sequence of commands, exceeding numbers of packets, or an non-typical combination of values in packet headers, is considered as anomalous.

An example of the communication pattern is depicted in Figure 3.8 that describes the activation of an IEC 104 device. The communication pattern includes four ASDUs exchanged between the RTU master and slave. Such pattern is directly extracted from ICS flow records as seen in Table 3.11. In Chapter 4 we will talk about an automated way how to extracted these communication patterns from ICS flow records and how to model them using probability automata.

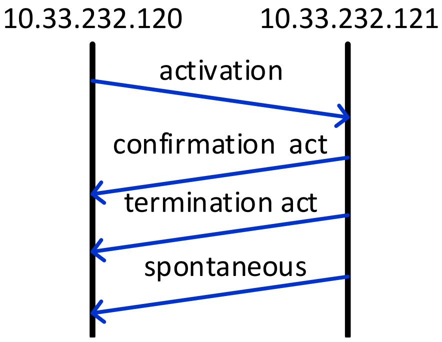
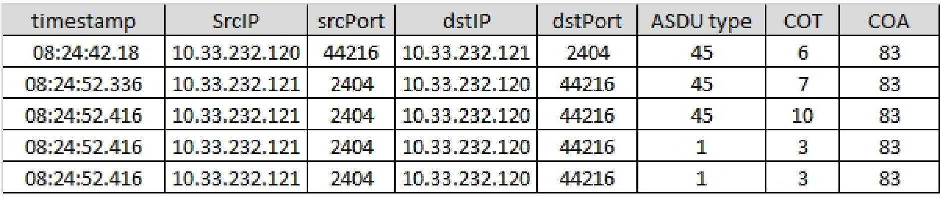


Figure 3.8: IEC 104 node activation command sequence



timestamp	SrcIP	srcPort	dstIP	dstPort	ASDU type	COT	COA
08:24:42.18	10.33.232.120	44216	10.33.232.121	2404	45	6	83
08:24:52.336	10.33.232.121	2404	10.33.232.120	44216	45	7	83
08:24:52.416	10.33.232.121	2404	10.33.232.120	44216	45	10	83
08:24:52.416	10.33.232.121	2404	10.33.232.120	44216	1	3	83
08:24:52.416	10.33.232.121	2404	10.33.232.120	44216	1	3	83

Table 3.11: IEC 104 node activation

Using historical ICS flow records we can also create a statistical model of smart grid communication. Statistical techniques have been considered as a resource efficient for anomaly detection techniques, see [100, 26, 37].

The statistical model is formally defined as a pair $(S, \mathcal{P})$, where:

- S is the sample space of the model that comprises the set of all possible tuples of features considered in the model, e.g., the number of ICS packets exchanged between ICS devices, their sizes, inter packet delay, direction, etc.

- $\mathcal{P}$ is a set of probability distributions on S.

The statistical model is then computed using the sample of ICS flows that represents normal behavior of the system as follows:

1. The flows are grouped in time windows of the predefined size, e.g., 60 seconds, 5 minutes, etc.

2. For each flow within the given window, selected features are extracted and added as a new tuple to the set of samples.

3. Step 2 is repeated until all windows are processed.

4. Finally, the set of probability distributions $\mathcal{P}$ is determined from all collected samples using a statistical inference method.

We demonstrate this approach on detection of the switching attack. We consider a simple method that computes a threshold to define the anomaly. In this simple case we will use only one feature: the number of transmitted packets within a time window. Figure 3.9 depicts a scenario where an attacker manipulates with the IED node by sending IEC 104 Activation (Act) and Activation Termination (ActTerm) commands within the ASDU.

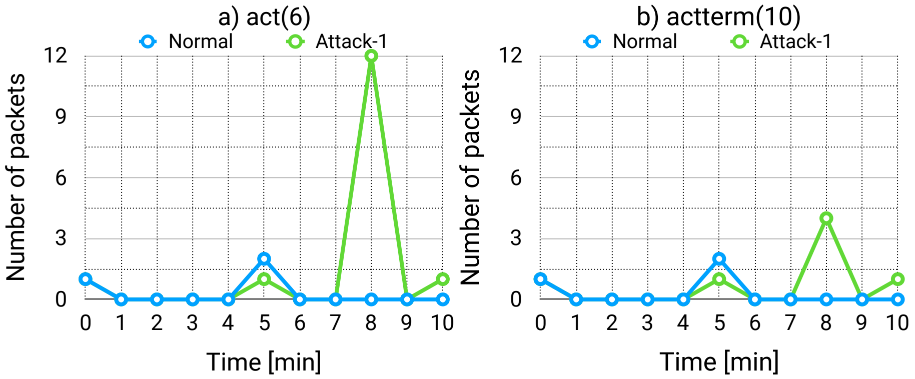


Figure 3.9: Distribution of (a) Act and (b) ActTerm packets during the normal usage and the attack.

The blue line represents the number of Act (a) and ActTerm (b) commands sent during 10 minutes of normal traffic. The green line depicts the number of Acts and ActTerms during the attack. We can see a peek between from 7 to 8 where unexpected number of Acts was sent to the IED device.

By comparing the number of sent Act commands with the number of received ActTerm commands we can notice that the number of Acts and ActTerms is equal during normal communication. This means that all Act ASDUs are correctly confirmed by the ActTerm as defined by the communication pattern in Figure 3.8. However, during the attack, some Act commands are ignored due to the high number of requests.

This scenario simulated behavior of the Industroyer malware, see Section 2.4.2, when an targeted IED device was continuously switched on and off within few seconds. Using ICS flow monitoring and simple statistics we are able to detect such behavior. More elaborated technique for statistical-based anomaly detection using ICS flows is given in Chapter 5.

3.7.1.3 Data Exfiltration Between ICS Devices and File Transfer

Data exfiltration describes an attempt to download data from an ICS system without the proper authorization. We mentioned in Section 2.3 that current ICS protocols lack security enhancement to prevent attacks against authorization, integrity, confidentiality, and availability. Despite this limitation, using ICS flow monitoring we can detect these kind of attacks.

Figure 3.10 depicts transmission statistics of a long term communication between IEC 104 nodes within 3 days. We can see the stable number of ASDUs transmitted over the network.

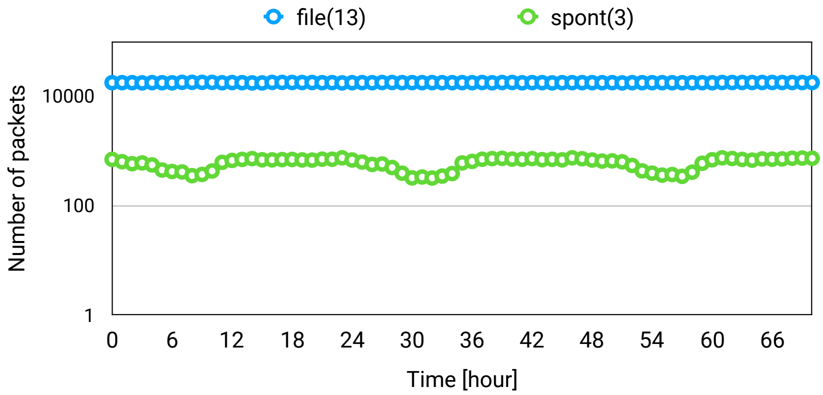


Figure 3.10: IEC 104 communication

The green line represents ASDUs with COT=3 (Spontaneous Event) sent in monitor direction to the RTU master. The blue line depicts the file transfer between two nodes (COT=13, Data Transmission). Using ICS flows, we can detect unauthorized data transfer by observing ICS flows with COT=13 and ASDU type 120-127 (File Transfer) and also determine the source and destination of such activity. Then, the operator should decide if this transmission was legitimate or not. However, without increased ICS visibility, such operation would not be noticed.

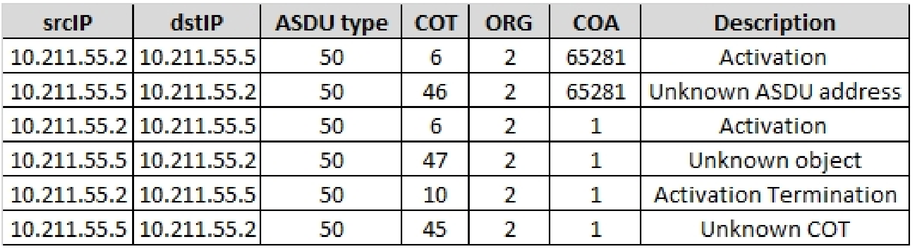
3.7.1.4 Resource scanning

Using ICS flow monitoring, we can detect not only new devices but also identify unknown or invalid resources. ICS device and port scanning (also called reconnaissance, see Section 2.3.2.1) is a preparatory phase before the cyber attack is launched. During this phase the attacker maps available resources on the network. A typical device scanning attack on the IP layer can be performed by `nmap` tool[6]. Some penetration tools provide resource scanning even on Layer 7, which is more difficult to detect.

Scanning attacks can be easily revealed by flow monitoring. IP flows identify the scanning attack by enumeration of IP addresses and ports that appears in flow statistics. Using ICS flows, we can also detect resource scanning on the Layer 7. Typically, the resource scanning attack yields a large number of packets targeting one device within a short time with invalid responses. By observing ICS fields that mark the invalid resource request, we can identify the attack.

[6] See https://nmap.org/ [July 2021]

In case of IEC 104 communication, we can observe COT values, see Table 3.12, especially values 46 (unknown address), 47 (unknown information object) or 45 (unknown Cause of Transmission). All these packets indicate either resource scanning attack or misconfiguration. In both cases, it is necessary to take some appropriate action.



srcIP	dstIP	ASDU type	COT	ORG	COA	Description
10.211.55.2	10.211.55.5	50	6	2	65281	Activation
10.211.55.5	10.211.55.2	50	46	2	65281	Unknown ASDU address
10.211.55.2	10.211.55.5	50	6	2	1	Activation
10.211.55.5	10.211.55.2	50	47	2	1	Unknown object
10.211.55.2	10.211.55.5	50	10	2	1	Activation Termination
10.211.55.5	10.211.55.2	50	45	2	1	Unknown COT

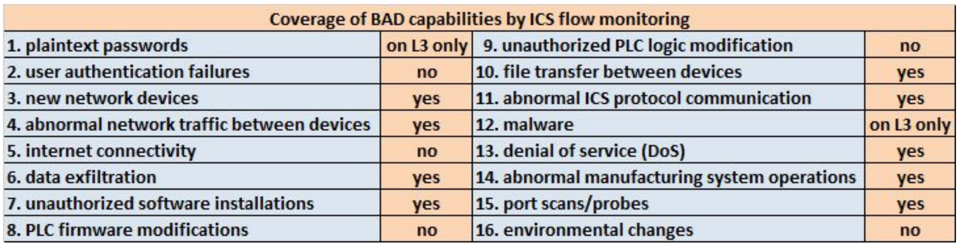
Table 3.12: IEC 104 wrong data

3.7.2 ICS flows and BAD capabilities

ICS flows provide a valuable source of monitoring data for successful detection of common security threats. Its design is based on a standardized IPFIX framework with ICS-specific templates, see Appendix B. Unlike IP flow monitoring, ICS flows provide additional details about ICS communication, which are essential for cyber threat detection.

ICS flow monitoring uses a passive approach which means that not all anomalies listed in Behavioral Anomaly Detection (BAD) classes [92], see Table 3.9, can be detected because it requires an agent running directly on a monitored device. For example, flow monitoring is not able to detect plain text password (BAD no.1) transmitted in ICS protocols, however, it is able to detect protocols that may transmit plain passwords, e.g., SMB, telnet or FTP. ICS flow monitoring cannot detect failed internet connectivity (BAD no.5) because this detection requires active testing. Also, it is not able to detect malware transmissions since it does not process packet payload. However, the system can recognize unusual communication patterns caused by malware activity, e.g., communication between the control station and an external site, which is typical for botnets.

Table 3.13 shows how ICS flow monitoring and analysis covers BAD capabilities. Majority of BAD capabilities is fully covered by ICS flow monitoring (value yes). BAD classes like plain text passwords detection or malware detection cannot be directly covered using ICS flows unless the ICS-enabled probe checks the content of transmitted ICS packets and informs about unencrypted passwords. Nevertheless, suspicious malware activities can be identified by analyzing Layer 3 transmissions that are part of ICS flow records. BAD defines also threats related to the hardware, e.g., BAD capabilities no.8 and 9, which are not subject of flow-based monitoring.



Coverage of BAD capabilities by ICS flow monitoring			
1. plaintext passwords	on L3 only	9. unauthorized PLC logic modification	no
2. user authentication failures	no	10. file transfer between devices	yes
3. new network devices	yes	11. abnormal ICS protocol communication	yes
4. abnormal network traffic between devices	yes	12. malware	on L3 only
5. internet connectivity	no	13. denial of service (DoS)	yes
6. data exfiltration	yes	14. abnormal manufacturing system operations	yes
7. unauthorized software installations	yes	15. port scans/probes	yes
8. PLC firmware modifications	no	16. environmental changes	no

Table 3.13: Coverage of BAD capabilities by ICS monitoring

3.8 Summary

Industrial systems are attractive targets for cyber criminals, activists, professional hackers, or state-sponsored attackers. Critical infrastructure is among the most significant concerns for cyber defense. Recent attacks against smart grids demonstrate the need to improve the security of smart grid communication. These attacks were not correctly detected due to an inadequate protection. To provide sufficient network security monitoring in ICS systems, visibility into communication is an essential requirement. Although many ICS systems use IP-based networking, standard enterprise security systems cannot analyze ICS application protocols, thus they are not able not to provide the required in-sight to network transactions.

In this chapter, we have introduced the concept of the ICS monitoring system employing IPFIX flows extended with application-level data extracted from ICS communication protocols. The approach was demonstrated on the IEC 104 communication, which is the standard protocol suite for smart grid networks. The proposed ICS flow monitoring is passive and does not affect network performance.

In order to mitigate security threats against smart grids, anomaly detection techniques can be integrated with the ICS flow-based network monitoring system. Thanks to ICS transaction visibility, it is possible to detect behavior anomaly detection cases as specified in the NISTIR 8219 report. We have demonstrated in Section 3.7 how ICS flow records help detecting rogue devices, abnormal network traffic, data exfiltration, and resource scanning. The simple statistical model was created based on the regular traffic to demonstrate this ability.

Following chapters present two solutions of automated anomaly detection of smart grid communication that are based on ICS flow monitoring data. The first high-level solution that models ICS communication sequences using probabilistic automata is presented in Chapter 4. The second solution that observes statistical distribution of packet features is described in Chapter 5. Both approaches provides a viable solution for improving security of network communication in the smart grid.

Chapter 4

Anomaly Detection Using Probabilistic Automata

Protection of the critical infrastructure against cyber attacks has become a challenge for security experts during recent years and was addressed by several security reports issued by NIST [113, 36, 92]. With adoption of IT technologies like TCP/IP or Ethernet, interconnection of industrial networks with Internet, remote access, etc., cyber threats against ICS systems increased. The cyber attacks initiated from the outside of the smart grid can be effectively filtered out on the perimeter of the network by ICS-enabled firewalls or IDS systems. This protection is, however, ineffective against attacks originating from the inside network. Such attacks can be launched by a malware installed on a control station, from a compromised host, or a rogue device connected to ICS/SCADA infrastructure as mentioned in Section 2.4. Attackers usually scan the ICS network first in order to identify potential targets. Then they launch a regular attack with the intention to control industrial processes [17], steal sensitive data [96], damage functionality of the system [16, 93], or request ransom for encrypted data [106].

In order to identify and eliminate internal cyber threats against ICS system, we need (i) to monitor ICS communication and (ii) to employ monitoring data for detection of suspicious behavior on the network. As showed in the previous chapter, high visibility of ICS communication can be achieved using IPFIX flow monitoring extended with meta data from the ICS protocol header. Monitoring data are stored in the ICS flow records that contain (i) flow properties, i.e., data extracted from Layer 3 to Layer 7 protocols, and (ii) statistical properties of the flow, i.e., the starting and ending time, the number of transmitted bytes, packets, see Section 3.3.1. Such data represent a valuable source of information about communication activities in the smart grid that is used for anomaly detection [121]. Flow records are usually collected at the network management system and then analyzed by an anomaly detection system (ADS) [56]. This process is depicted in Figure 4.1. The most important question is how to model ICS conversations obtained from ICS flow records. This chapter presents a solution based on formal languages and automata.

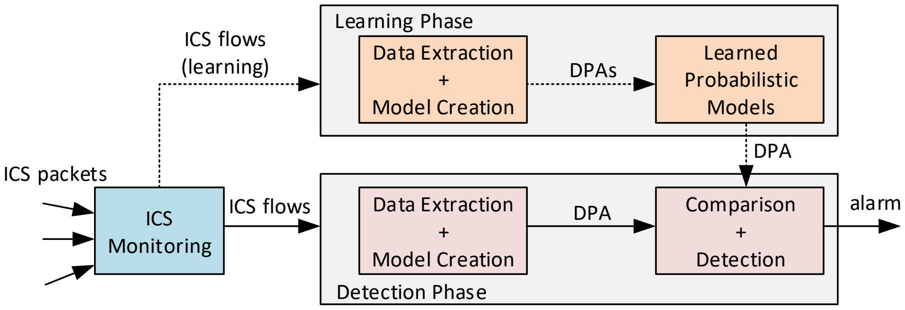


Figure 4.1: Monitoring and detecting anomalies using ICS flows

The idea behind the proposed approach stems from the observation that ICS traffic between two ICS devices is stable and predictable [21, 19, 102]. In addition, a set of commands that are regularly exchanged between these device is relatively small and can be expressed using a formal language. Having ICS flow records with commands, we can create a multiset of ICS conversations. This set together with the frequency of command occurrence represent a set of strings belonging to a probabilistic language that describes ICS communication exchanged between two devices. In this chapter we describe how to create a model of this language using probabilistic automata [39] and how we can detect anomalies using this model.

The chapter is structured as follows. First, we overview research works that deal with modeling ICS communication. Then we introduce theory of probabilistic automata developed by Colin de la Higuera [39]. We show how to create a probabilistic model from ICS flow records. The model can be represented by a Prefix Trees (PT) or a Deterministic Probabilistic Automata (DPA). We use these models to describe a normal communication of the ICS network. Then we present two anomaly detection techniques that are applied on our probabilistic models: *single conversation reasoning* and *distribution reasoning*. Finally, we evaluate these techniques on IEC 104 communication and discuss their advantages and limitations.

The work presented here was developed in collaboration with Vojtěch Havlena and Lukáš Holík from the Department of Intelligent Systems FIT BUT in frame of Bonnet Project[1].

4.1 Related Work

Anomaly detection (AD) of ICS/SCADA communication has been explored by many research teams in previous years as a response to the increasing threats of cyber attacks against the critical infrastructure [92, 102]. Unlike signature-based approach where detection systems search for suspicious sequences in individual packets, anomaly detection creates a model of the legitimate behavior of an ICS

[1] See https://www.fit.vut.cz/research/project/1303/.en [July 2021].

system during normal operations. Then, ADS system observes deviations of an input traffic with respect to the normal behavior model. If the deviation is higher then a given threshold, the input communication is marked as anomalous.

Rakas et al. [102] divide anomaly detection systems into three groups: statistical-based (univariate, multivariate, time series), knowledge-based (finite automata, description scripts, expert systems), and machine learning-based (using Bayesian networks, Markov models, neural networks, fuzzy logic, etc.). Our approach is a combination of knowledge-based and machine learning-based techniques because we employ probabilistic approach as in Markov models and use finite automata to implement the model.

Similar approach was explored by Lin and Nadjm-Tehrani [78, 79] who observed three attributes of IEC 104 communication (ASDUTYPE, COT, IOA) and created a Probabilistic Suffix Tree (PST) that represented underlying timing patterns of spontaneous events for each attribute class. Using the changes in distribution of inter-arrival times, they categorized the traffic into five different groups based on the periodicity and stability of observed times. They used PSTs to predict the future behavior of the network and detect possible changes. Their method is computationally very demanding and also sensitive to network delays. Instead of modeling timing features, our technique models sequences of ICS commands exchanged between communication nodes with their probabilities and is able to detect unexpected operations or higher occurrence of some commands.

Martinelli et al. [83] proposed a network of timed automata (TA) to model the SCADA water distribution system. They mapped numerical values of water tank level into three classes. Time changes represented edges in the time automata. They implemented anomaly detection using formal verification of pre-defined temporal logic formulae over this model. However, their method had several limitation, mainly manual creation of TAs and then high computational requirements during model checking.

Goldenberg and Wool [49] similarly to us modeled semantics of ICS protocol, more specifically, sequences of queries and responses of Modbus communication. Their model employed Deterministic Finite Automata (DFA) where symbols of the alphabet represent a tuple of a transaction ID, function code, reference number, and bit/word count obtained from the Modbus packet. DFA transitions then expressed the predicted behavior of the system which could be of type normal, retransmission, miss, or unknown. The created model was sensitive to out-of-order messages. Similar to us, their model was able to recognize invalid messages. States of our automaton also express expected commands but transitions represent probability of next expected command. Thus, our model is able to comprehend more aspects of communication, e.g., frequency of exchanged commands, the correct order of command sequences, etc.

Probabilistic approach to SCADA communication was also applied by Caselli et al. [27, 28] who introduced a sequence-aware intrusion detection system based on Discrete-Time Markov Chains (DTMC). The modeling process clusters all messages with the same semantic meaning to one state of the DTMC, e.g., read coils from address 0 for Modbus, Initiate for MMS, or type ”I, C_IC_NA_1” for IEC 104. Transitions represent a sequence of messages using the probability of moving from one state (a message) to another state (a subsequent message). Transitions also include information about the number of jumps and average time between two consequent states. Our model is different. We do not cluster similar messages into one state but each sequence of messages is modeled as a string of the language which produces more accurate model. Thus, we are able to detect changes in network behavior by observing ICS communication. In addition, our model is created automatically using the set of ICS conversation without the need to specify exact semantics of each command.

The idea of inferring protocol state automata from network traces was also explored by Wang et al. [123] where the authors created a Probabilistic Protocol State Machine (P-PSM) from network communication traces of the given protocol. P-PSM is a probabilistic generalization of protocol state machine. First, the authors measured similarity between messages using Jaccard index and clustered similar messages into groups using Partitioning Around Medoids (PAM) algorithm. Their definition of P-PSM is very similar to Deterministic Probabilistic Automata (DPA) defined by de la Higuera [39]. Their protocol model is more general since the similar messages are grouped into one state, e.g., commands EHLO and HELO for SMTP are considered as one cluster. This is sufficient for observing typical behavior of a specified protocol. Our approach is more accurate since each each sequence of messages become a part of the model without any clustering. This helps to identify even small changes in communication.

Similar approach of learning stateful protocol models from network traces was examined by Kreuger et al. [74] who developed a method for protocol inspection and state machine analysis. They represented messages by *n-grams* and model communication states by a hidden Markov model. They used the model for simulating behavior of honeypots. As stated in the previous paragraph, our approach is different because we do not model a specific protocol behavior but communication exchanges, even if both approaches use probabilistic model.

To our best knowledge we can say that the proposed automata-based approach of modeling communication sequences for anomaly detection is novel and have not been published by other authors. It differs from the previously published methods and gives promising results in anomaly detection as demonstrated later.

4.2 Preliminaries

A language is a set of strings over defined an alphabet where a string either belongs to a language or does not. The problem of defining a language from a set of sample strings have been studied by many researchers in the past. One of the early works related to this domain was introduced in 1987 by Dana Angluin who presented a technique to identify an unknown regular set from examples of its members and nonmembers [14]. She developed a learning algorithm $L^\star$ that correctly learns any regular set from any minimal adequate *teacher* in polynomial time. This system requires the teacher to determine whether string t belongs to the language or not (membership query). Then the teacher is asked if a conjectured automaton is equivalent to the target language (equivalence query). In a stochastic setting the teacher can be replaced by a random sampling oracle that determine membership of a sample. Application of the $L^{\star}$ algorithm to our domain is not possible because we learn the language from the samples only and have not a teacher that would decide the equivalence question.

Later, automata learning have been studied by other researchers [42, 62, 15]. Similarly to Angluin's idea, the proposed algorithms require a piece of knowledge that decides membership and equivalence queries whether a given string belongs to the target language.

Our problem with modeling ICS communication is different. We have a regular set of samples obtained from the normal ICS traffic. Since all these samples belong to the unknown language, we do not need to ask for membership as the above mentioned algorithms suggest but we are not able to determine if a constructed automaton is complete since there is no teacher that would answer the equivalence questions. Thus, for our specific domain is more appropriate to construct probabilistic automata that do not require a teacher. Instead of defining a language as a set of strings, we deal with a distribution over a set of strings [39]. The distribution can be regular, in which case the strings are then generated by a probabilistic regular grammar or a probabilistic finite automaton. The following text introduce the notion of probabilistic automaton and probabilistic distribution that is important for our method of anomaly detection.

4.2.1 Probabilistic Automata[2]

Let Σ be a finite alphabet and Σ^* the set of all finite strings over alphabet Σ, with ϵ denoting the empty string.

Definition 4.1. A Probabilistic Automaton (PA) is a tuple $\mathcal{A} = (\Sigma, Q, \delta, \mathbb{I}, \mathbb{F})$ where

- Σ is the alphabet,
- Q is a finite set of *states* labeled $q_1, \ldots, q_{|Q|}$,
- $\delta : Q \times \Sigma \times Q \rightarrow Q \cap [0, 1]$ is a (total) *transition function* assigning probabilities from the interval $[0, 1]$ of rational numbers to *transitions*,
- $\mathbb{I} : Q \rightarrow Q \cap [0, 1]$ is a mapping assigning the *initial-state probabilities*,
- $\mathbb{F} : Q \rightarrow Q \cap [0, 1]$ is a mapping assigning the *acceptance probabilities*.

[2] The following definitions are mostly taken from [39].

The probabilistic automaton must satisfy the *consistency* condition requiring that for each state q, the sum of probabilities of the outgoing transitions plus the probability of acceptance is 1, that is $\forall q \in Q$:

$$\mathbb{F}(q) + \sum_{a \in \Sigma, r \in Q} \delta(q, a, r) = 1$$

Definition 4.2. A Probabilistic Automaton $\mathcal{A} = (\Sigma, Q, \delta, \mathbb{I}, \mathbb{F})$ is called *deterministic* (DPA) if the following conditions are satisfied:

- $\exists q \in Q$ (unique initial state) such that $\mathbb{I}(q) = 1$,
- $\forall q \in Q, \forall a \in \Sigma : |\{r \mid \delta(q, a, r) > 0\}| = 1$, that is, for each $q \in Q$ and $a \in \Sigma$ exist a unique successor.

Further, we define probability of acceptance of a word w by probabilistic automaton $\mathcal{A}$.

Definition 4.3. Let $\mathcal{A} = (\Sigma, Q, \delta, \mathbb{I}, \mathbb{F})$ be a probabilistic automaton and $w = a_1 \dots a_n \in \Sigma^*$ be a word. A *trace* π of the word w is a sequence $\pi = (q_0, a_1, q_1) \dots (q_{n-1}, a_n, q_n)$ where $\delta(q_{i-1}, a_i, q_i) > 0, \mathbb{I}(q_0) > 0$ and $\mathbb{F}(q_n) > 0$ for $1 \leq i \leq n$. Informally, it is a path through PA via w starting in the initial state and ending in a state with non-zero acceptance probability.

Definition 4.4. *Probability of trace* π is then given as $\mathcal{P}_{\mathcal{A}}(\pi) = \mathbb{I}(q_0).\delta(q_0, a_1, q_1), \dots, \delta(q_{n-1}, a_n, q_n).\mathbb{F}(q_n)$. Informally, we multiply transition probabilities with the initial-state probability and the acceptance probability on trace π.

Let $\prod_w$ be a set of all traces of word w. *Probability of word* w accepted by $\mathcal{A}$ is the sum of probabilities of all traces for the given word, that is,

$$\mathcal{P}_{\mathcal{A}}(w) = \sum_{\pi \in \prod_w} \mathcal{P}_{\mathcal{A}}(\pi)$$

Example 4.1. Consider a PA from Fig. 4.2. Then $\mathcal{P}_{\mathcal{A}}(abc) = 1.0 \cdot 0.3 \cdot (0.2 \cdot 0.3 + 0.5 \cdot 0.1) = 0.033$.

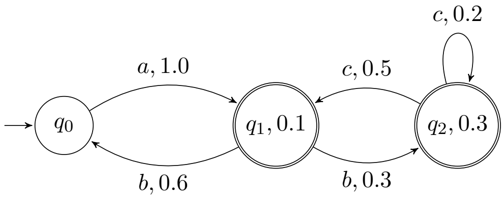


Figure 4.2: Example of a probabilistic automaton. States are labeled with a state name and the accepting probability (no number corresponds to zero probability). Transitions are labeled with a symbol and the probability taking this transition.

4.2.2 Frequency Automata

For construction of DPA, we need to define a deterministic frequency finite automaton (DFFA) that is later used to form a DPA.

Definition 4.5. A *Deterministic Frequency Finite Automaton* (DFFA) is a tuple $\mathcal{A} = (\Sigma, Q, \delta_{fr}, \mathbb{I}_{fr}, \mathbb{F}_{fr})$ where

- Σ is an alphabet,
- Q is a finite set of states,
- $\delta_{fr} : Q \times \Sigma \times Q \to Q \cup \mathbb{N}$ is a (total) transition frequency function. Due to determinism, for each $q \in Q$ and $a \in \Sigma$ there is at most one state $q\prime \in Q$: $\delta_{fr}(q, a, q\prime) > 0$,
- $\mathbb{I}_{fr} : Q \to \mathbb{N}$ is a mapping assigning initial-state frequencies. Because of deterministic automaton, there is exactly one state $q : \mathbb{I}_{fr}(q) > 0$,
- $\mathbb{F}_{fr} : Q \to \mathbb{N}$ is a mapping assigning final-state frequencies.

The *consistency* condition of DFFA says that any string that enters a state (or starts in a state) has to leave it (or end there). If we denote by FREQ(q) the total number of both entering and leaving strings at state q, the consistency condition is formally expressed as follows:

$$\begin{aligned} FREQ(q) &= \mathbb{I}_{fr}(q) + \sum_{r \in Q, a \in \Sigma} \delta_{fr}(r, a, q) && (4.1) \\ &= \mathbb{F}_{fr}(q) + \sum_{r \in Q, a \in \Sigma} \delta_{fr}(q, a, r). && (4.2) \end{aligned}$$

If the consistency condition is fulfilled an DFFA can be *normalized* to an DPA by dividing the acceptance frequencies of each state q and frequencies of its outgoing transitions by its overall frequency $FREQ(q)$ as demonstrated at the example showed at Figure 4.3.

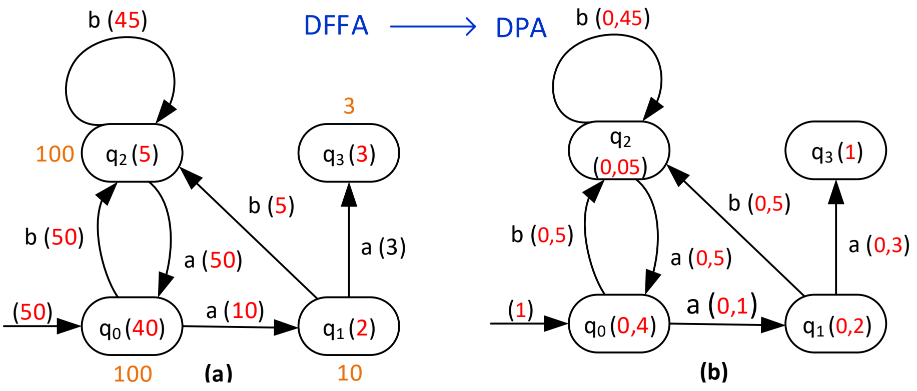


Figure 4.3: Transformation of DFFA to DPA.

Frequency values of DFFA at Figure (a) are in brackets. This DFFA fulfills the consistency condition, so it can be transformed to DPA. The transformation algorithm is straightforward and uses $FREQ(q)$ values computed for each state of the DFFA. These values are marked by yellow color in the figure. For example, state q_0 of DFFA contains one incoming edge from q_2 and $\mathbb{I}_{fr}$ with the total frequency 100. We get the same sum for outgoing frequencies plus the finish-state frequency $\mathbb{F}(q_0)$. Transition algorithm defined in [39, pp. 390-391] computes the DPA's probabilities by dividing the DFFA's frequencies by $FREQ(q)$ value, i.e.,

$$\mathbb{F}(q) = \frac{\mathbb{F}_{fr}(q)}{FREQ(q)}, \delta(q,a,r) = \frac{\delta_{fr}(q,a,r)}{FREQ(q)}.$$

Note that for deterministic PA $\mathbb{I}(q) = 1$, see Definition 4.2.

4.2.3 Prefix Tree

For learning probabilistic language from samples, we use a special type of the DFFA called a Prefix Tree (PT). Before defining the Prefix Tree, we need to define a prefix set of language L.

Definition 4.6. The *prefix set* of L is $PREF(L) = \{u \in \Sigma^\star : uv \in L\}$.

We use the $PREF(L)$ for defining the prefix tree. The prefix tree is a compact representation of the input multiset S. Its nodes are prefixes of strings in S where ϵ is the root and there is an edge labeled by symbol v from u to $u.a$ if and only if both u and $u.a$ are prefixes of strings from S. The edge is labeled by the number of occurrence of prefix $u.a$ in S. We denote the number of occurrence of string w in S as $S(w)$:

$$\sum_{w \in S, \exists v: w = u.a} S(w).$$

Formally, the prefix tree is defined as follows:

Definition 4.7. Let S be a multiset of strings from $\Sigma^\star$. The *Prefix Tree* PT(S) is the DFFA:$(\Sigma, Q, \delta_{fr}, \mathbb{I}_{fr}, \mathbb{F}_{fr})$ where

- $Q = \{q_u : u \in PREF(S)\}$,
- $\mathbb{I}_{fr}(q_\epsilon) = |S|$ (initial-state frequency for q_ϵ),
- $\forall u.a \in PREF(S), \delta_{fr}(q_u, a, q_{u.a}) = |S|_{ua\Sigma^\star}$ (transition frequency of $u.a$),
- $\forall u.a \in PREF(S), \delta_{fr}(q_u, a) = q_{u.a}$ (the next state for the sample $u.a$),
- $\forall u \in PREF(S), \mathbb{F}_{fr}(q_u) = |S|_u$ (final-state frequency).

An example of the prefix tree generated from a set of 100 strings is showed in Figure 4.4. The values in brackets represent frequencies: initial-state, final-state, or transition frequencies.

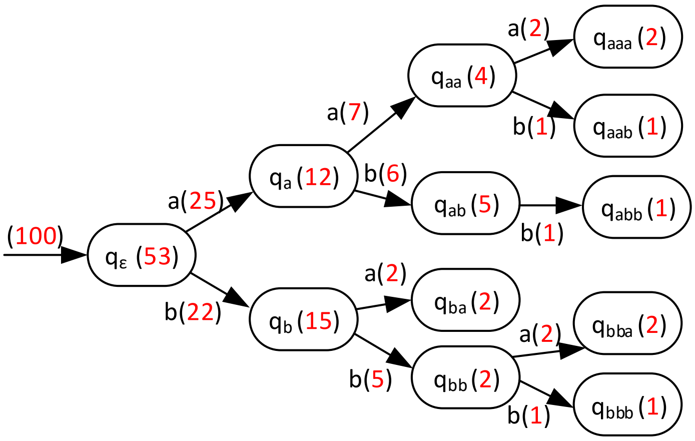


Figure 4.4: Example of the Prefix Tree.

From the prefix tree, we can see that a part of the input set are two strings aaa, bba, one string aab, abb, bbb, four string aa, five string ab, etc. Thus, from a set of input samples, we can easily create a prefix tree representing this set of samples. The prefix tree is a special form of DFFA, so it can be transformed to a DPA which is more compact form for probabilistic language representation. The following section shows how PTs and DPAs can represent ICS communication.

4.3 Modeling ICS Communication Using Automata

In this section we apply the theory of probabilistic automata as presented in the previous section on ICS communication. First we explain, how prefix trees or DPAs can be created from input samples. Then we describe steps how ICS flow records are transformed into a multiset of conversations for modeling ICS traffic using probabilistic automata.

4.3.1 From Samples to Probabilistic Automata

For learning the DPA we use algorithm Alergia presented in [39]. Here, we briefly outline how the algorithm works.

Given a multiset S of input strings, the algorithm creates a prefix tree of these strings with the frequency of string occurrence in the multiset. Then the prefix tree is transformed into the deterministic probabilistic automaton that approximates the probabilities of the individual strings in S by merging similar states. The algorithm proceeds in the following steps:

1. Create prefix tree PT(S) with strings from S where each edge is labeled by the frequency of occurrences of the respective string prefix in S, see Section 4.2.3. The prefix tree is created iteratively:

(a) For each string $w \in S$ we check if exists a trace $\pi(w)$ over the tree. If yes, we increase the frequency of occurrence $S(w)$ along the trace $\pi(w)$ by one.

(b) If trace $\pi(w)$ does not exist, we find the shortest prefix $PREF(w)$ that is already a part of the tree. For the rest of string w we add new states and transitions according to the Definition 4.7.

Prefix tree $PT(S)$ may be interpreted as a frequency automaton of S where nodes are states, edges correspond to transitions, ϵ is the initial state, and the acceptance frequency of each state w equals $|S(w)|$.

2. Generalize and compact the PT by merging ”similar” states.

   Generalization is the main part of Alergia. This algorithm includes ordering of the states, a compatibility test and merging and folding operations, see [52] for details. The general algorithm visits the states through two loops and attempts to merge states recursively. It explores the prefix tree from the initial state. While exploring the tree, it merges states r on the frontier of the so far undiscovered part of the tree with the previously discovered states q.

   The merging operation merges the sub-tree rooted by r into the automaton reachable from q. The acceptance frequency of r is added to the acceptance frequency of q. Moreover, for each symbol a, the frequency of the outgoing a-transition of r is added to the frequency of the outgoing a-transition of q, and the merging procedure is recursively called on the target states of the two merged transitions.

   Two states q and r are merged under the condition that they are sufficiently similar. Similarity here means that their acceptance frequencies are close enough as well as the frequencies of the outgoing a-transitions for each symbol a. The test that is used to decide if states are to be merged or not is based on the Hoeffding test made on the relative frequencies of the empty string and of each prefix. The quantities compared are (we use f_1, f_2, n_1, n_2 for shortening):

$$\frac{f_1 = \mathbb{F}_{fr}(q)}{n_1 = FREQ(q)} \text{ and } \frac{f_2 = \mathbb{F}_{fr}(r)}{n_2 = FREQ(r)}.$$

   Then the comparison test using the Hoeffding's bounds is implemented as follows:

$$\text{Test}(f_1, f_2, n_1, n_2, \alpha) = \left|\frac{f_1}{n_1} - \frac{f_2}{n_2}\right| < \left(\sqrt{\frac{1}{n_1}} + \sqrt{\frac{1}{n_2}}).\sqrt{\frac{1}{2}.\ln\frac{2}{\alpha}}\right) \tag{4.3}$$

3. The generalized PT (which is a DFFA) is then transformed to the corresponding DPA as demonstrated on Figure 4.3.

The level of similarity in Step 2 is controlled by parameter α that corresponds to the probability that the merged automaton wrongly rejects a string from S. The authors of the algorithm recommends $\alpha = 0.05$. In our preliminary experiments we set $\alpha \in \{0.05, 0.1, 0.15, 0.2\}$. We discovered that the best results for our datasets is given by $\alpha = 0.05$ which is also the recommended value by the authors of the algorithm. Detailed description of the algorithm and our experiments is given in the technical report [52].

In our experiments, we use two models of the input multiset: the Prefix Tree created from the input multiset by Step 1, or the Deterministic Probabilistic Automaton, created by Steps 1 to 3. In Section 4.5, we evaluate these two representations using our datasets and conclude how suitable they are for anomaly detection.

4.3.2 ICS Flows Data Pre-Processing

Now we describe how to pre-process ICS flow records and create a sample set S representing ICS communication for learning probabilistic automata.

Collecting ICS flows. ICS flows are obtained from the ICS-enabled monitoring probe as described in 3.3. The probe observes passing traffic and creates ICS flow records that contain selected data from Layer 3 to 7 headers. In case of IEC 104 communication, the ICS flows contain the following L7 headers: APDU type, ASDU type, Number of information elements, Cause of Transmission (CoT), Originator address (ORG), and ASDU address (COA). For IEC 104 protocol modeling, we observe only two IEC 104 headers: ASDUTYPE and COT.

Partitioning the traffic by communication groups. Given network flows, our aim is to obtain an automaton for each *communication group*, i.e., a set of communicating ICS devices. For ICS protocols employing *master-slave* communication mechanism, the communication group usually includes two devices (master and slave) identified by a pair of SrcIP:srcPort + DstIP:dstPort. Another communication topology used in SCADA systems includes a set of slaves (RTUs) that are requested by one master station. This is typical for MODBUS but it is also used for IEC 104. In this communication model we observe ICS messages exchanged between the master and slave(s) in both directions.

For ICS protocols with *publish-subscribe* mechanism (e.g., GOOSE) the communication group includes a publisher with all subscribers. The communication group is identified by the publisher ID and destination ID, e.g., the publisher's source address and subscriber's destination multicast address. The traffic of the communication group includes all messages sent by the publisher (one-directional communication).

Thus, we partition the observed traffic according to communication groups. Since we use ICS flow monitoring, ICS packets are represented by a set of ICS flow records of all devices in the communication group observed within a given time window.

Splitting the traffic into conversations. The learning algorithm from Section 4.3.1 takes a multiset of strings as an input. Network traffic of the communication group is represented by a sequence of ICS flow records. For automata-based modeling, we need to divide ICS flow records into a multiset of *conversations* which are sequences of logically connected messages that correspond to one ”communication session”. The result of splitting phase is a sample set S of the learned conversations that is used for constructing a probabilistic model: a prefix tree or DPA.

Recall that we work with messages on the application layer. Thus, there can be multiple ICS conversations within one TCP session which is typical, for example, for IEC 104 protocol. Recall the conversation is a sequence of messages logically bound together. A splitting of a sequence of messages into conversations is done automatically according to the semantics of ASDUTYPE and COT. A conversation is built by adding messages from a traffic one by one until a specific suffix or prefix is spotted.

Example 4.2. Identification of a conversation in the sequence of flow records is based on the expert knowledge of the particular ICS protocol. Since the conversation is a logical sequence of L7 messages, we need to define what ICS messages logically close the conversation. Based on IEC 104 protocol standard [85], we specify the following suffixes and prefixes that split IEC 104 communication into conversations:

- Suffixes: IEC 104 conversions are typically ended by messages with ASDUTYPE = 70 (end of initialization), 124 (ACK file), and packets with COT = 9 (confirmation deactivation), 10 (confirmation termination), or 44–47 (unknown resources), and also messages transmitted as single units, e.g., COT=3 (spontaneous event).

- Prefixes: Start of an IEC 104 conversation can be marked by the following messages: COT=4 (initialized), ASDUTYPE=120 (file ready) or 122 (select file).

If logical sequences of ICS messages (i.e., conversations) are not properly determined for a given ICS protocol, e.g., because of a lack of expert knowledge, the original conversations can be cut into several parts and from the input ICS flow records we obtain a different sample set that is later used for learning. This will result in a different automaton. However, this automaton describes the same ICS communication (expressed by ICS flow records) and thus, it can be used for anomaly detection. Disadvantage of the improper splitting of input data is that the semantics of obtained conversations do not reflect the real meaning of transactions and can be confusing for experts.

Message abstraction. To represent normal network communication using automata, we need to set a suitable level of abstraction and remove irrelevant details from messages. Too much details would lead to an over-specialized model that marks small nuances in communication as anomalies while too little details would blur the boundaries between normal communication and anomalies.

- First, the level of message abstraction depends on available ICS flow data. We discussed four levels of ICS visibility in Section 3.6. Depending on the level of ICS visibility, we select appropriate ICS flow headers that sufficiently describe behavior of ICS communication with respect to the specifics of a given ICS protocol. The set of recommended ICS headers for common ICS protocols in smart grids is given in Table 3.1.

- Second, it is necessary to consider semantics of the selected field and decide how unique the values are for probability distribution modeling. For instance, each message includes a timestamp which makes the message unique. Thus, the learning procedure could hardly find any regular structure in communication and the learned automaton would hence reject all messages appearing in the future as anomalies.

  In case of IEC 104 protocol, the ASDU contains a set of information objects and information elements, see Figure 3.4. Modeling individual information objects and elements would provide interesting details about communication, on the other hand it would lead to a very large multiset of input samples with large probabilistic model. Since we primarily aim to model the exchange of ICS commands, we consider command-level abstraction only. This abstraction reveals ICS operations on the network without going into specific details of each operation. If a detailed analysis of ICS traffic is required, there is a possibility of full packet capturing. In Section 4.4 we demonstrate, that command-level abstraction is sufficient for anomaly detection of most cyber attacks against smart grid communication.

For IEC 104 protocol [85], we particularly consider fields AsduType and CoT that determine the high-level communication model and abstract from concrete data values. An abstract ICS message is modeled as a pair ⟨AsduType, CoT⟩, see the following example.

Example 4.3. Consider a sample of conversations S consisting of a sequence of pairs ⟨AsduType,CoT⟩, i.e.,

```
S = { [<122,13>, <120,13>, <122,13>, <121,12>, <122,13>, <125,13>, <125,13>,
       <125,13>, <123,13>, <124,13>, <123,13>, <124,13>],
      [<36,3>],
      [<36,3>],
      [<36,3>],
      [<122,13>, <120,13>, <122,13>, <125,13>, <123,13>, <124,13>],
      ...
    }
```

The incomplete prefix tree and the DPA constructed from this sample set is depicted in Figure 4.5.

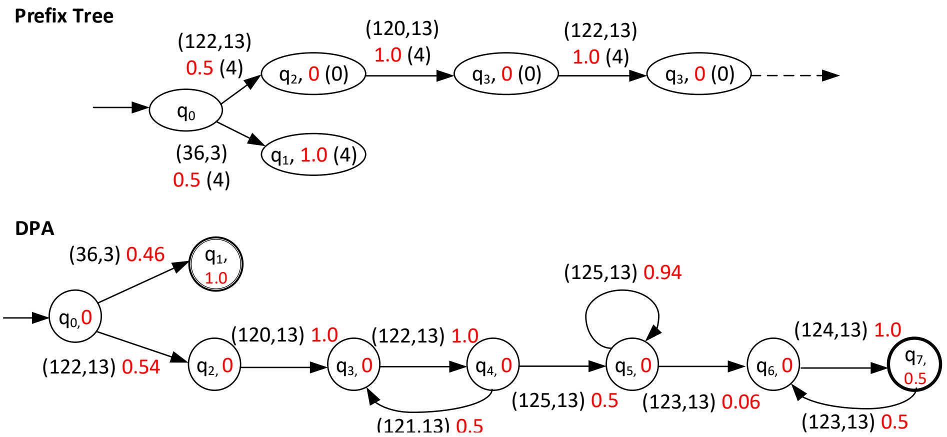


Figure 4.5: Learning a prefix tree and DPA from IEC 104 samples.

This model reveals what operations are typically executed between these two IEC 104 communicating devices. If a testing traffic contains a different communication sequence, it is considered as anomaly.

We can summarize data pre-processing by three steps, see Figure 4.6:

1. *Partitioning:* from a given dataset, e.g., IEC 104 communication, extract only the IEC 104 flow records with i-messages and partition them by pairs of communication entities.

2. *Splitting:* Split the modified traffic into conversations.

3. *Abstraction:* Select suitable fields from ICS messages that create an abstract communication model.

The set of abstracted conversations for each communication group becomes an input for probabilistic model learning as described in Section 4.3.1.

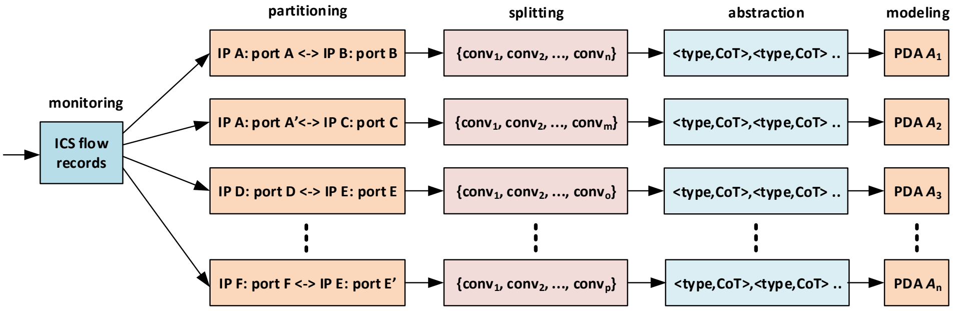

partitioning
splitting
abstraction
modeling
monitoring
ICS flow records
IP A: port A <-> IP B: port B
IP A: port A'<-> IP C: port C
IP D: port D <-> IP E: port E
IP F: port F <-> IP E: port E'
{conv1, conv2, ..., convn}
{conv1, conv2, ..., convm}
{conv1, conv2, ..., convo}
{conv1, conv2, ..., convp}
<type,CoT>,<type,CoT> ..
PDA A1
PDA A2
PDA A3
PDA An

Figure 4.6: Pre-processing ICS flow data.

4.4 Anomaly Detection

Now we present how to utilize the learned model of a network traffic for anomaly detection. Our detection works with fixed time windows (particularly, 5 minutes) where each time window contains a set of conversations as described above. Optionally, flexible sliding time windows can be implemented, which does not have impact the principle of presented anomaly detection mechanisms.

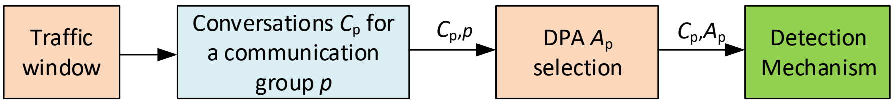


Figure 4.7: Overview of the anomaly detection.

Detection has three consecutive phases as also depicted in Figure 4.7:

1. The time window is divided into a series of conversations $\mathcal{C}_p$ for each communication group p identified, e.g., by the end-to-end IP addresses and ports.
2. A probabilistic automaton A_p that was created during the learning phase and describes the normal communication of p is selected using communication group identifiers.
3. Anomalies are detected by comparing $\mathcal{C}_p$ with $\mathcal{A}_p$.

The last step, anomaly detection based on a comparison of $\mathcal{C}_p$ and $\mathcal{A}_p$, is implemented by the *Single Conversation Reasoning* or by the *Distribution Reasoning* as explained in the following text.

4.4.1 Anomaly Detection via Single Conversation Reasoning

The first mechanism for anomaly detection is based on reasoning about individual conversations. For each conversation $c \in \mathcal{C}_p$, we compute the probability $\mathcal{P}_{\mathcal{A}_p}(c)$ (see Definition 4.4) assigned to c by probabilistic automaton $\mathcal{A}_p$ that represents valid communication of group p that was created during the learning phase. If the probability is below the threshold μ, i.e., $\mathcal{P}_{\mathcal{A}_p}(c) \leq \mu$, an anomaly is detected. For our detection, we set threshold μ to 0 meaning that we are only interested in whether $\mathcal{A}_p$ marks c as possible string of the language (no matter how far), or not. If $\mathcal{P}_{\mathcal{A}_p}(c) = 0$, conversation c is not accepted by $\mathcal{P}_{a_p}$ and is marked as anomaly.

The advantage of single reasoning mechanism is that it is simple, fast and allows to point to the concrete conversation causing the anomaly which provides useful data for further diagnostics. The time complexity of single conversation reasoning is linear and depends on an input sample set, i.e., $\mathcal{O}(|\mathcal{C}_p|.l)$ where $|\mathcal{C}_p|$ is the number of input samples and l is the length of a longest string from $\mathcal{C}_p$.

One major limitation of the single conversation reasoning is that this method is not able to detect missing conversations, e.g., when a link or device go down. If conversation c is in the learning set but it is not present in the testing set, we cannot compute $\mathcal{P}_A(c)$ for the missing conversation c, so this kind of failure will not be detected. This case is, however, covered by the distribution reasoning described in the following section.

4.4.2 Anomaly Detection via Distribution Comparison

Instead of evaluating individual conversations gathered from the monitoring traffic in isolation as in case of the single conversation reasoning, the second detection mechanism focuses on evaluating a traffic transmitted in a fixed-length time window. The probabilistic distribution of every window is then compared to the probabilistic distribution of the learned model expressed by automaton $\mathcal{A}_0$, see Figure 4.8. Unlike the single conversation reasoning, this mechanism also detects anomalies caused by missing conversations or by a change of a communication profile.

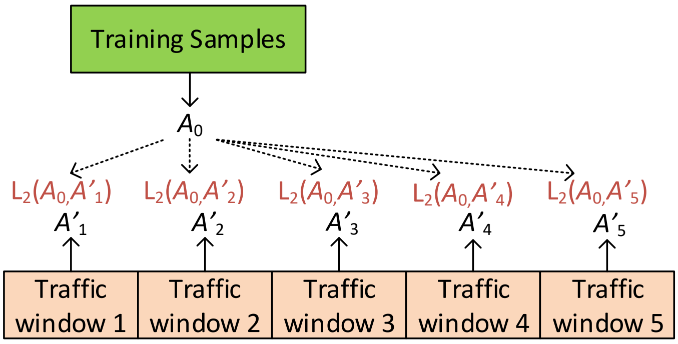


Figure 4.8: Distribution Comparison Reasoning.

The idea of this approach is to learn a DPA for testing traffic $\mathcal{C}_p$ and then compare DPA $\mathcal{A}_{p\prime}$ with the referential DPA $\mathcal{A}_p$ representing the valid traffic. The detection mechanism works as follows:

1. Construct a DPA $\mathcal{A}'_p$ from a sequence of conversations $\mathcal{C}_p$ obtained from the traffic window under monitoring.
2. Compare the $\mathcal{A}'_p$ with the DPA $\mathcal{A}_p$ representing the normal traffic. If the difference is too large, report an anomaly.

To quantify how much different is $\mathcal{A}_p$ from $\mathcal{A}'_p$, we use the 2-Euclidean distance L_2 (or just Euclidean distance), defined as

$$L_2(\mathcal{A}_p, \mathcal{A}'_p) = \sqrt{\sum_{w\in\Sigma^*} \Big(\mathcal{P}_{\mathcal{A}_p}(w) - \mathcal{P}_{\mathcal{A}'_p}(w)\Big)^2} \tag{4.4}$$

Intuitively, the Euclidean distance sums the differences of probabilities assigned to strings accepted by $\mathcal{A}_p$ and $\mathcal{A}'_p$.

In [39], the authors also mention other metrics for measuring distance between automata: L_1 distance (variation, Manhattan distance), logarithmic distance, or Kullback-Leibler divergence. Logarithmic distance is useful when considering very small probabilities which is not our case. One drawback of Kullback-Leibler divergence is that it represents a relative metric where the sum over all strings of the logarithmic loss is weighted down by the actual probability of a string. If any string has a null probability in $\mathcal{A}_{p'}$ but not in $\mathcal{A}_p$, then the denominator is null and the Kullback-Leibler divergence is infinite. There are some ways how to solve this issue but since this case may appear in our domain, we prefer to use L_2.

Time complexity of this algorithm is polynomial, more specifically $\mathcal{O}(n^6)$, where n is the maximum number of states of $\mathcal{A}_p$ and $\mathcal{A}'_p$. For efficient implementation of probabilistic automata we use matrix representation, see [119] for details.

To reduce false positives we use a parameter θ to control if these two automata are different enough to announce an anomaly, i.e., $L_2(\mathcal{A}_p, \mathcal{A}'_p) > \theta$. The value of θ represents sensitivity of detection on the interval $[0, 1]$. Lower value of θ means higher possibility of false alarms, higher values can cause that some anomalies would not be discovered. Based on our experiments we recommend θ values from 0.1 to 0.25.

4.5 Experiments

In this section we present our experiments with modeling and anomaly detection of IEC 104 communication using prefix trees and probabilistic automata. In the first part we focused on learning and will discuss how efficient are PTs and DPAs for representing the smart grid communication. The second part applies probabilistic models on datasets with selected cyber attacks where we evaluate efficiency and accuracy of anomaly detection.

4.5.1 Learning Probabilistic Models from IEC 104 Samples

For modeling IEC 104 communication, we employed the Alergia algorithm presented in Section 4.3.1. For our first experiments, we used tool Treba[3] developed by Mans Hulden et. al [58]. The tool implements various algorithms for training probabilistic finite automata including Alergia. For execution, the Treba tool uses two parameters: α that determines if two states have frequencies close enough to be merged, see Equation (4.3), page 67, and threshold t_0 that determines the minimal number of strings that are necessary to be a state considered for merging [39, p. 400]. Based on our experiments [52], we set the parameters as follows:

- The parameter α is set to 0.05 which gives a good balance between the merging (the strength of generalization and compactness) and classification error.

[3] See https://code.google.com/archive/p/treba/ [August 2021]

Dataset	IEC 104 flows	i-messages	Conv.	Devices
iec104	115	91	31	2
10122018-104Mega	104,533	94,040	6,927	4
10122018-104Mega_0	9,905	8,876	503	2
13122018-mega104	1,460,829	1,313,997	91,957	14
13122018-mega104_1	62,040	55,772	3,603	2
mega104-14-12-18	14,597	9,657	9,125	2
mega104-17-12-18	58,930	37,661	37,661	2
KTH-RTU1	6,234,474	3,117,251	2,088,540	6
KTH-RTU1_1	184	96	59	2
KTH-RTU1_2	168	87	55	2
KTH-RTU4	3,306,086	1,653,046	1,107,537	2
RICS	1,550,304	775,152	519,352	2

Table 4.1: IEC 104 datasets used for experimental evaluation

- The threshold t_0 is set according to the formula $t_0 = \lfloor \log_2 |S| \rfloor$, i.e., the threshold depends on the size of the sample set. The logarithmic function was chosen to obtain a smaller increase with the growing number of samples.

We evaluated the proposed probabilistic models on real IEC 104 traffic[4]. The characteristics of our datasets (name, the number of flows, i-messages, conversations, and communicating devices) are summarized in Table 4.1. Our datasets contain from 31 to millions of conversations. The number of devices occurring in the traffic varies between 2 and 14. Datasets with more than two devices are partitioned by communication groups and one of the partitions is selected for experiments (parts are annotated with numbers, e.g., 0,1,2). We also include the full unpartitioned version which represents one master more slaves communication.

We applied the learning algorithm Alergia on each dataset to get the corresponding prefix tree model and DPA. One third of each dataset was used for learning, the other two thirds were used for testing, i.e., evaluating the accuracy of the learned model. The accuracy was computed as the ratio of the accepted conversations (with non-zero probability) to all conversations in the testing data. The results are shown in Table 4.2.

The table evinces a high accuracy of both DPA and prefix tree models (about 99%) in all cases except iec104 dataset. The case of iec104 illustrates a scenario with an insufficient learning data. The learning sample contains only one third of the 115 messages and 31 conversations, which does not cover the complexity of overall communication. Thus, the learned model has a very little chance to recognize the testing communication. Notice, that in this case Alergia is not able to merge states and generalize, so it returned an automaton with the same size of 44 states as the prefix tree.

[4] Datasets created in BUT are available in CSV format at https://github.com/matousp/datasets/ [Sept 2020]. Datasets KTH-RTU1, KTH-RTU4, and RICS were provided by the RTSLab, Linköping University, Sweden [78].

Dataset	**DPA learned by Alergia algorithm**			**Prefix tree**	
	Est. parameters	*States*	*Accuracy*	*States*	*Accuracy*
`iec104`	$\alpha = 0.05,\ t_0 = 3$	44	0% (0/31)	44	0% (0/31)
`10122018-104Mega`	$\alpha = 0.05,\ t_0 = 11$	8	100% (4642/4642)	49	99.8% (4636/4642)
`10122018-104Mega_0`	$\alpha = 0.05,\ t_0 = 7$	8	99.7% (337/338)	48	99.7% (337/338)
`13122018-mega104`	$\alpha = 0.05,\ t_0 = 14$	8	99.9% (61606/61612)	38	99.9% (61606/61612)
`13122018-mega104_1`	$\alpha = 0.05,\ t_0 = 10$	8	99.9% (2414/2415)	28	99.8% (2412/2415)
`mega104-14-12-18`	$\alpha = 0.05,\ t_0 = 11$	8	100% (6114/6114)	39	100% (6114/6114)
`mega104-17-12-18`	$\alpha = 0.05,\ t_0 = 13$	3	100% (25233/25233)	3	100% (25233/25233)
`KTH-RTU1`	$\alpha = 0.05,\ t_0 = 19$	12	100% (2088540/2088540)	12	100% (2088540/2088540)
`KTH-RTU1_1`	$\alpha = 0.05,\ t_0 = 4$	9	98.3% (58/59)	9	98.3% (58/59)
`KTH-RTU1_2`	$\alpha = 0.05,\ t_0 = 4$	9	100% (55/55)	9	100% (55/55)
`KTH-RTU4`	$\alpha = 0.05,\ t_0 = 19$	10	100% (1107537/1107537)	10	100% (1107537/1107537)
`RICS`	$\alpha = 0.05,\ t_0 = 17$	2	100% (519352/519352)	2	100% (519352/519352)

Table 4.2: Modeling IEC 104 communication using DPAs and prefix trees.

For some cases, e.g., `13122018-mega104` and `10122018-104Mega`, an application of Alergia lead to a slightly small number of false positives that are resulted by messages that were wrongly classified as anomalies. In case of `10122018-104 Mega`, we get 100% accuracy for Alergia and 99.8% for the prefix tree. This is caused by the fact that Alergia merges the prefix tree and derive general regularities. Thus, the constructed DPA represents an over-approximated model of the input set. Due to over-approximation (or generalization) the DPA also accepts conversations that do not precisely appear in the learning sample which decreases the number of false negatives.

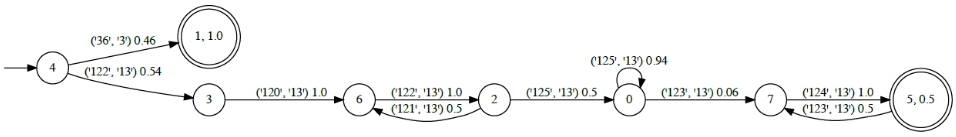


Figure 4.9: A DPA constructed from dataset `13122018-mega104_1`

.

In this particular case, Alergia learned that the file transfer may contain any number of data segments, i.e., messages with ASDUTYPE=125 and COT=13, see Figure 4.9. Thus, the model classifies as normal also conversations which are contained in the learning sample and have the different number of occurrence. On contrary, the prefix tree classifies everything that did not appear in the learning sample as anomaly because it skips the generalization phase. The number of false positives generated by the prefix tree is, nevertheless, quite small (below 2%). This can be explained by the fact that we deal with a highly regular and relatively simple traffic which is almost covered by the learning sample.

The prefix tree learned from `KTH-RTU4` dataset is depicted in Figure 4.10. Note that the probabilities contain rounded values, therefore the probability denoted as 0.0 means a very small value, e.g., $1.8 \cdot 10^{-6}$ for transitions from 6 to 0 and 6 to 9.

As seen from our experiments, Alergia creates more compact automata than the

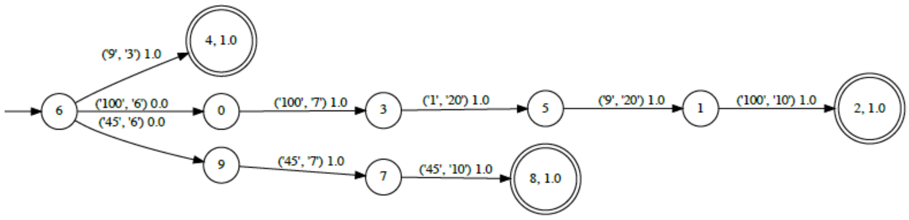


Figure 4.10: A prefix tree constructed from dataset `KTH-RTU4`.

prefix tree due to merging of similar states in the generalization phase. The number of states created by the prefix tree is surprisingly small, despite the large size of the learning set. It is because of relative simplicity of ICS communication. For a couple of datasets, in particular `KTH-RTU*`, the prefix tree has the same number of states as the automaton obtained by Alergia. The reason of this is nature of these datasets where the traffic is almost indifferentiable to apply the state merging.

The advantage of the prefix tree over the DPA created by Alergia is its simplicity and transparency. For highly regular traffic it has relatively small number of states with respect to the size of the learning set.

4.5.2 Anomaly Detection Using Probabilistic Models

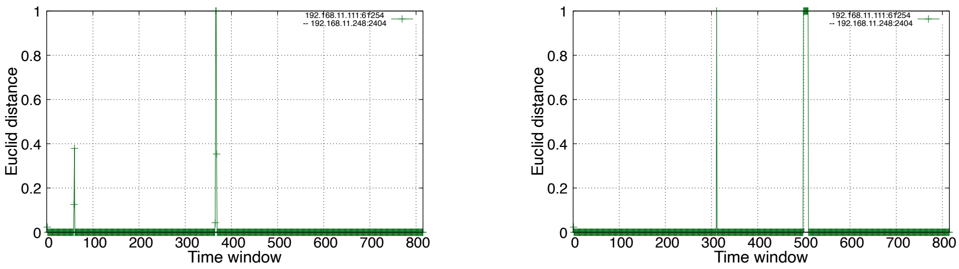
In previous text we presented a method how to model ICS traffic using probabilistic models. In this part, we focus on evaluation of our anomaly detection mechanisms mentioned in Section 4.4, particularly single conversation reasoning and distribution comparison reasoning.

For our experiments we used IEC 104 dataset `mega104-17-12-18` that was created at Brno University of Technology[3]. The dataset consists of 58,930 messages of IEC 104 communication that were captured within 3 days of a real network traffic. We experimented with six types of anomalies discussed below. The attack traces were inspired by real scenarios against ICS communication as described in Section 2.3 and 2.4. These attack vectors were emulated by injecting/removing IEC 104 messages into/from the original normal traffic while keeping all the IEC 104 features untouched.

The probabilistic model of the normal traffic was trained on the original traffic. The results of anomaly detection using DPA were indistinguishable from results when using prefix trees for learning. Therefore we give only one common summary of the results based on DPAs.

Input data were analyzed within five minutes windows. The output of distribution reasoning using DPA is visualized in Figures 4.11, 4.12, and 4.14.

Injection attack. In this scenario, see Fig. 4.11 a), an attacker compromised an internal host on the ICS network and started to send unusual requests. First, the attacker sent activation messages with ASDUTYPE=45 and COT=6 which requested execution of the given command on the target host. The host correctly



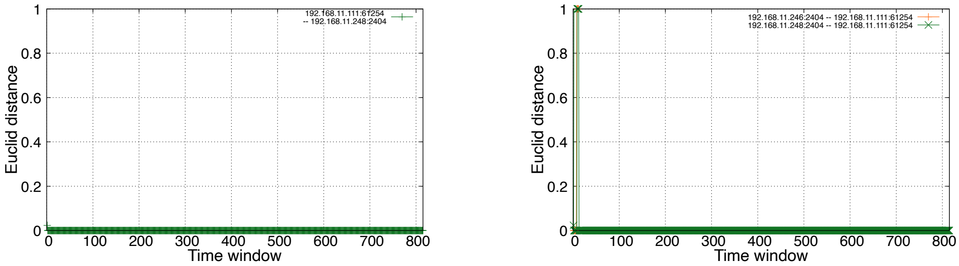
(a) Injection attack

(b) Connection loss

Figure 4.11: Anomaly detection using distribution reasoning, part I.

confirmed the request with CoT=7. The first attack took five minutes and included 83 packets. During the second injection attack the attacker transferred a file from the target to the compromised host. The attacker sent messages with AsduType $\in \{122, 120, 121, 124, 125\}$ which represented a file transfer. The attack included 221 messages and took 15 minutes. We can see that during the first attack, the Euclidean distance was about 0.4, during the second attack it was 1.

Connection loss. This scenario, see Figure 4.11 b), represents a short blackout of a device when connection was lost. The first connection failure took 10 minutes and 146 messages were lost. The second failure lasted for about one hour and 921 messages were lost. In both cases is the connection loss successfully detected by distribution reasoning over DPAs.



(a) DoS attack

(b) Rogue devices

Figure 4.12: Anomaly detection using distribution reasoning, part II.

DoS attack. This Denial of Service (DoS) attack, see Figure 4.12 a), was directed against a control station. The attacker sent hundreds of legitimate packets to the destination. He used a spoofed IP address, which was sending spontaneous messages with AsduType=36 and CoT=3. The attack lasted for half an hour and contained about 1049 spoofed messages. As seen in Figure 4.12 a), the attack was not detected. It is because the DoS attack scenario contained additional conversations of the same type A that was present in the training dataset. The time windows of the valid communication also contained many conversations of the type A and the constructed probabilistic automata could not capture the change.

Example 4.4. To make it clear, consider, for instance, a time window containing 10 messages of the type A and another time window containing 1.000 messages of type A. Then probabilistic automata obtained from the prefix tree and these time windows are equal, see Figure 4.13.

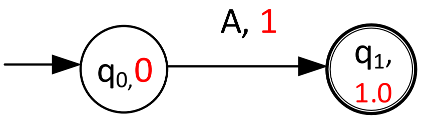
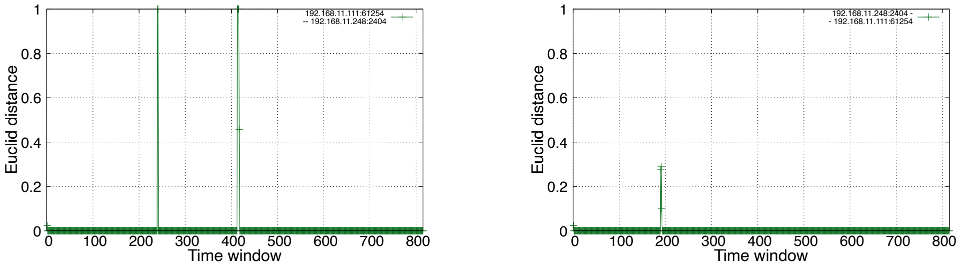


Figure 4.13: A DPA created from A sample set.

This limitation of probabilistic automata can be removed by a combination of the detection procedure with a simple statistical analysis.

Rogue devices. A rogue device was connected to the ICS network and started communicate with an IEC 104 host using legitimate IEC 104 messages. The attacker used a sequence of spontaneous messages with ASDUTYPE=36 and COT=3. The station correctly responded with supervisory APDUs. The attack lasted about 30 minutes and included 417 packets, see Figure 4.12 b). As seen in the figure, the attack was correctly identified.



(a) Scanning attack

(b) Switching attack

Figure 4.14: Anomaly detection using distribution reasoning, part III.

Scanning attack. This scenario includes two scannings: the horizontal scanning that enumerated IP addresses of the network segment, and the vertical scanning that enumerated IOA addresses on the selected host, see Figure 4.14a a). First, the attacker sent IEC 104 Test Frame messages on port 2404 that is typical for IEC 104 and observed responses. If a station responded, the attacker launched the vertical scanning of the host using General Interrogation ASDUs sent to information objects with IOA addresses 1 to 127. Each attack took about 15–20 minutes and was detected by the distribution reasoning method.

Anomaly	*Single*	$Distr_{pref}$	$Distr_{aler}$
Communication loss	✗	✓	✓
Switching attack	✓	✓	✓
Scanning attack	✓	✓	✓
DoS attack	✗	✗	✗
Rogue devices	✓	✓	✓
Injection attack	✓	✓	✓

Table 4.3: Comparison of the detection methods

Switching attack. The switching attack implemented a scenario similar to the real attack against Ukrainian power plant using CrashOverride malware [41]. During this attack a series of IEC 104 packets with ASDUTYPE=46 and a sequence of COT numbers $(6, 7, 10)$ were sent to the target that caused switching the device on and off, see Figure 4.12 b). The attack lasted for ten minutes and transferred 72 packets. Also here the attack was detected because such sequence of IEC 104 packets was not present in the normal ICS traffic and cause anomaly expressed by higher Euclidean distance of DPAs.

4.5.3 Discussion

We evaluated our detection methods described in Section 4.4 using cyber attacks scenarios typical for smart grid networks. For detection using the single conversation reasoning we set the threshold $\mu = 0$. For detection via distribution comparison we set $\theta = 0.25$. The length of an observed time window was five minutes which corresponds to a typical export time of IPFIX records.

Results of both detection method, i.e., single conversation reasoning and distribution reason are summarized in Table 4.3. We have compared the detection via single conversation reasoning (*Single*), detection via distribution comparison based on DPAs learned using Alergia ($Distr_{aler}$), and detection via distribution comparison based on prefix trees ($Distr_{pref}$). The detection results for $Distr_{aler}$ of the considered scenarios are shown in Figures 4.11 to 4.14. The graphs show Euclidean distance of the valid traffic and the traffic under inspection for each time window, see Equation 4.4 in Section 4.4.2.

From Table 4.3 we can see that the $Distr_{aler}$ and $Distr_{pref}$ detection methods are equally successful in all cases except the DoS attack scenario as discussed above.

The *Single* detection method does not detect DoS attack and communication loss. In case of communication loss, *Single* is not able to detect an anomaly because it only analyses existing individual conversations unlike the distribution comparison method.

From graphs in Figures 4.11 to 4.14 we can see that in case of $Distr_{aler}$, we are able to detect all anomalies including multiple occurrence within the scenarios (except the discussed DoS attack scenario) with no false positives. The same is true also for $Distr_{pref}$: the graphs look the same, so we do not present them here. In case of the *Single* detection approach, the situation is also encouraging. This detection method is able to detect all anomalies including their multiple occurrences.

Our detection methods do not report any false positives because no other windows in the traffic are evaluated as anomalous. They give alerts exactly on the ongoing anomalies, except the two missed anomalies discussed above.

4.6 Summary

In this chapter, we have introduced a new technique for efficient modeling of ICS communication using probabilistic automata. Our choice of probabilistic automata as a modeling mechanism for the network traffic is based on the idea that DPAs can be efficiently learned from positive examples and that besides the regular structure of the communication, the probabilistic automata also capture a probability distribution of communication which is beneficial, for instance, for detection of connection loss.

Since the ICS communication is stable and regular, the automata capture the normal communication rather precisely using small number of states and transitions. The automata are automatically learned from samples of ICS communication obtained from ICS flow records. The automata model the semantics of ICS communication between two ICS devices. The semantics is extracted from the protocol headers based on expert knowledge. We demonstrated on IEC 104 communication that it is enough to consider only ASDUTYPE and Cause of Transmission (COT) extracted from i-messages for modeling high-level communication of IEC traffic.

For anomaly detection we proposed two methods that verify incoming data in form of ICS flow records with the learned probabilistic model. The first method called *single conversation reasoning* computes the probability of each incoming conversation on the learned automaton. If the probability is zero, the conversation is marked as anomalous. The second method called *distribution comparison reasoning* creates a probabilistic model for each time window of the input traffic. This model in form of the prefix tree or DPA is compared to the learned model. Difference between model is expressed using Euclidean distance. If the distance between these two models is above threshold, an anomaly is raised.

We demonstrated that these detection methods are able to detect common classes of cyber attacks on ICS systems, i.e., the switching attack, command injection, connection of a rogue device, and the scanning. Probabilistic automata are not suitable for detecting denial of service attacks if the attack uses same communication sequences that were present in the training dataset. However, the DoS can be easily detected by statistical methods as described in the following chapter.

Chapter 5

Statistical-Based Anomaly Detection

In the previous chapter we introduced an automata-based approach for modeling ICS communication sequences using probabilistic automata. As demonstrated by our experiments, the created automata-based models are stable, compact and able to detect security threats and anomalies typical in smart grids. We also noticed that due to the missing timing information, an automata-based model cannot capture time-related anomalies like an abundance of legitimate ICS packets caused by the DoS attack, unexpected inter-packet delays which may indicate unstable connection, or irregular time delivery caused by network congestion. It is possible to extend probabilistic automata by time but their application to ICS traffic would be limited due to their complexity. Instead, we focus our attention on statistical methods that produce sufficient results for ICS anomaly detection with much lower time and space complexity then probabilistic timed automata.

To address time-related issues of smart grid communication, we thus propose a second approach of modeling ICS communication. This approach can be seen as complementary to the automata-based modeling where automata represent high-level qualitative features of ICS communication while statistical-based modeling deals with quantitative communication parameters and their statistical distribution.

Statistical-based anomaly detection observes distribution of specific features obtained from ICS traffic. By monitoring ICS packets and flows, we extract various features like inter-arrival time, packet direction or size that are used to build a model representing their statistical distribution. Anomaly detection then compares the real-time traffic statistics with the learned statistical model. Data points that are outside the learned model are called outliers or anomalies and denote unexpected behavior of the system.

Due to the stability of ICS communication, statistical models represent a natural way for detection of common ICS anomalies including security threats, device malfunctioning, network congestion, etc. In addition, monitoring time properties of ICS/SCADA protocols is extremely important to secure proper operation of critical industrial processes which work in real-time environment.

Statistical properties of network communication are not limited to time values only but include various packet and flow features that are observed on different layers of the TCP/IP model. On Layer 3 (IP layer), we can observe IP datagram timestamp, packet size, direction, delay, etc. On Layer 4 (transport layer), we can monitor TCP flow duration, flow size, TCP segment inter-arrival time, round-trip time (RTT), or re-transmissions [46, 69]. Statistically interesting features can be also extracted from Layer 7 (application layer), e.g., statistical distribution of IEC 104 commands (i-frames) or Modbus operations [71].

The main advantage of the statistical model is that it does not require high processing power and time to extract packet features and build the model, so it can be easily implemented on a monitoring probe or IDS device. On the other hand, statistical methods are sensitive to outliers which are particular data with exceptionally low probability that can be incorrectly marked as anomalies. Hence, an important question for statistical modeling of network communication is how to represent statistical distribution of a given data set so that the model is precise enough and includes even samples with low probability, and at the same time is able to correctly detect any anomaly. The level of detection accuracy is usually controlled by a threshold variable which is determined with respect to the specific environment.

In this chapter we present a method for creating a statistical profile of ICS communication using inter-arrival time and packet direction obtained from normal traffic samples. The presented method combines the ideas used by Crotti et. [37] and [108] for IP traffic fingerprinting and classification. We split the communication into regions based on packet inter-arrival times and direction. For each region we create a statistical profile representing the typical ICS traffic. The main point of this method is how to determine split-points that automatically divide the traffic into regions for more accurate modeling. For detection, we employ the Three Sigma Rule [101] that gives minimum false positives. Our experiments were mostly done on IEC 104 communication where we detected common anomalies. The proposed method not only detects a specific anomaly but is able to identify its type.

The work presented here was created in collaboration with Ondřej Ryšavý and Ivana Burgetová in frame of Ironstone[1] and Bonnet[2] projects.

5.1 Related Work

Statistical-based anomaly detection is one of the widely used techniques [12, 45]. The basic idea of statistical methods is to detect significant deviations of observed behavior from the normal one. Successful statistical modeling requires stable and predictable behavior of modeled traffic. Stability and regularity of ICS/SCADA

[1] See https://www.fit.vut.cz/research/project/1101/.en [July 2021].

[2] See https://www.fit.vut.cz/research/project/1303/.en [July 2021].

communication was previously studied and demonstrated for major industrial protocols like Modbus [117], IEC 104 [78], or DNP3 [46].

In our own work [89] we observed regularity of Internet of Things traffic and created a simple statistical model for representing resource usage of Constrained Application Protocol (CoAP) [112]. The CoAP resource was described by a pair *operation* (e.g., PUT command) and *resource URL address* (e.g., floor_light). In each time window we observed the number of packets and octets associated with the resource and created a usage profile related to the specific resource and device. The model was created by application of expectation-maximization (EM) algorithm [40] and represented as a joint probability function and computed threshold value. The obtained results showed hit ratio (recall) about 75 to 90% with false positive ration about 2 to 6.4%. Since ICS traffic is more stable and regular than CoAP communication, we applied a statistical model with simpler computation which, however, gives quite precise results.

Statistical properties of ICS communication were widely explored by Barbosa, et al. in [18, 20] where the authors compared periodicity, throughput and topology changes in SCADA and SNMP traffic. Their results show that SCADA communication exhibits periodic behavior at a smaller scale, has constant throughput over a long period of time, and keeps a stable number of connections. Its periodicity is caused by a polling mechanism used to retrieve data from SCADA slaves [19]. The authors demonstrated that attacks like scanning, denial of service, network protocol manipulation, or buffer overflow disturb the periodicity, thus, it can be detected by anomaly detection. For modeling the SCADA communication, Barbose et al. use time series representing the number of packets belonging to a specific flow. During periodicity learning, they generate a periodogram for each flow using Fast Fourier Transform. In detection phase, using discrete-time Short-Time Fourier Transform they create a spectrogram for monitoring changes in periodicity. Our approach comes out of Barbosa's observations. Instead of monitoring a simple number of transmitted packets we provide a more subtle classification using arrival times distribution. This is faster in computation while providing similar results.

Valdes and Cheung [117] introduced pattern-based and flow-based anomaly detection of ICS communication. Their patterns include source and destination IP addresses and ports. During detection, they monitor previous n-occurrences of the pattern and compute the historical probability of the pattern. If the probability is less than the given threshold, an alert is generated. Their solution includes a periodic update of the patterns and pruning the rare patterns. The second technique presented by Valdes and Cheung uses flow records for anomaly detection. Flow records include more attributes like source and destination addresses, the time of the last packet, the average number of bytes per packet, the variance of bytes per packet, or mean and variance of packet inter-arrival time. Similarly to pattern-based detection, they compare the traffic with historical flow records and compute a difference. If a record does not exist or differs too much, the alert is raised. They tested the approach on MODBUS network with periodic data retrieval. They were able to detect anomalies like scanning, modified data, denial of service, and system degradation. Unfortunately, their results do not show the number of false positives and implementation. Our approach does not observe individual flows but creates a model for entire communication between groups of communicating ICS nodes.

Lin and Nadjm-Tehrani [78] analyzed timing patterns of spontaneous events of the IEC 104 protocol which are asynchronously generated by an RTU. The authors model inter-arrival times of IEC 104 packets using Probabilistic Suffix Trees (PSTs) and analyze phase transitions, predictability, and frequent patterns. They describe inter-arrival times as sequences of symbols representing groups of ”similar” inter-arrival times. The symbolic sequences are further processed (smoothing, finding boundaries) and used to create a PST. Having the PST, the authors define a phase transition, i.e., a period of time during which the distribution of inter-arrival times is stable. They found five groups of traffic patterns based on phase transitions: strongly cyclic, weakly cyclic, stable, bursty, and transitional communication. Using the probability of communication patterns, they predict future behavior, i.e., that a certain pattern would appear in the next segment. The approach is, however, computationally very intensive. We also deal with IEC 104 communication, but we do not restrict to spontaneous events only but model all IEC 104 packets. We use a simpler statistical model with lower computational requirements.

In their other work, Lin et al. [79] propose a timing-based anomaly detection system for SCADA networks where they employ inter-arrival time of packets similarly to us. They built a statistical model for selected packets of three ICS protocols: request and responses of S7, requests and responses of Modbus, and IEC 104 spontaneous events. Their model includes sampling distribution defined by the sample mean, standard deviation, and the Central Limit Theorem (CLT). For detection, they use a sliding window where they calculate the sample mean and sample range. They verified the proposed model on normal traffic and various attacks including flooding, injection, and prediction (spoofing). They reached a 99% detection rate with 1.4% false positives. In our case, we divide packets into several regions based on inter-arrival time and direction, and for these regions we create a statistical model which is more accurate.

5.2 Preliminaries

Statistical anomaly detection is grounded on the assumption that ”normal data instances occur in high probability regions of the stochastic model, while anomalies occur in the low probability regions” [29]. Statistical modeling is a popular technique for anomaly detection because statistical methods allow simple and fast outlier detection, especially in one-dimensional space. They assume data points to be spread out according to some distribution, e.g., normal distribution. Then, a statistical test is performed in order to determine if a particular data point belongs to the model or not. If the probability of a particular data point generated from the learned model is low, then the data point is declared as an anomaly. Similarly, it is possible to define an interval of normal values for a given dataset using the statistical model and the applied test.

There is a various range of statistical methods and techniques used for anomaly detection. In this section, we recapitulate only those methods and statistics that are relevant for our work.

5.2.1 Sample Mean, Standard Deviation, Range

Having a sample of data points expressed using numerical values, we used several statistics that summarize the information in the sample data and highlights important features such as the middle or central tendency and the variability which is important for anomaly detection. There are three common statistics used for statistical modeling: the *sample mean*, *standard deviation* and *sample range* [94].

Let $D = \{d_1, ..., d_n\}$ be a sample of data points that follow the normal distribution. For set D, we compute the sample mean m, sample standard deviation σ, and the sample range r as follows:

$$\text{sample mean: } m = \frac{1}{n}\sum_{i=1}^{n} d_i \tag{5.1}$$

$$\text{sample standard deviation: } \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - m)^2} \tag{5.2}$$

$$\text{sample range: } r = max(d_i) - min(d_i) \tag{5.3}$$

5.2.2 Three Sigma Rule, Box Plot

A simple outlier detection technique is *Three Sigma Rule*. It says that for normal distribution roughly 99.7% of data points lie within the interval $\langle m - 3 \times \sigma, m + 3 \times \sigma\rangle$ [101] where m is the mean and σ is the standard deviation of the sample. This interval describes normal behavior and is computed during the learning phase. When a data point is outside this interval, it indicates an anomaly.

Another useful technique is the *Box Plot Rule* [116] that defines an interval of normal values using the Inter Quartile Range (IQR). Having the lower quartile Q_1 and upper quartile Q_3 of the distribution, the normal values are within the interval $\langle Q_1 - 1.5 \times IQR, Q_3 + 1.5 \times IQR\rangle$ where $IQR = Q_3 - Q_1$. For normal distribution, roughly 99.3% of data points lie within this interval.

In our research, we applied both approaches on our data [24]. Since the IQR test produced more false positives we focused on the Three Sigma Rule.

5.2.3 Packet inter-arrival time

Packet inter-arrival time Δt is *the amount of time between the arrival of two subsequent packets*. It is computed by a monitoring probe as a difference between timestamps of two subsequent packets. Its value depends on the location of the monitoring probe in the network, see Figure 5.1.

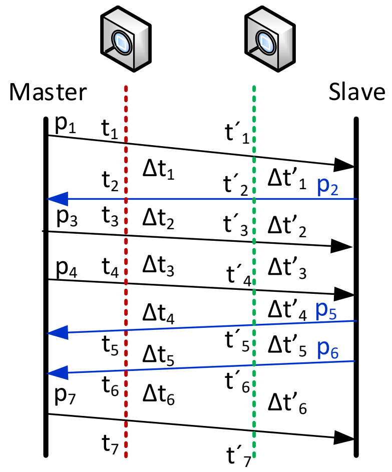


Figure 5.1: Measuring inter-arrival times

For statistical modeling, the location of a probe is not important since Δt distribution keeps the same statistical properties independently on the location of an observation point, i.e., statistical distribution of set $T = \{\Delta t_1, \Delta t_2, \Delta t_3, \Delta t_4, \Delta t_5, \Delta t_6, \ldots\}$ is similar to distribution of $T' = \{\Delta t'_1, \Delta t'_2, \Delta t'_3, \Delta t'_4, \Delta t'_5, \Delta t'_6, \ldots\}$. Nevertheless, the location of the probe during learning phase where a distribution model is created should be the same as for the testing.

In case of industrial communication and anomaly detection, it is effective to observe distribution of inter-arrival times between two packets in one direction or between packets in bi-directional transmissions.

In case of one-directional inter-arrival times monitoring, we observe independently packets passing through an observation point in each direction, see Figure 5.2 (a). Thus, we obtain two distributions of inter-arrival times sent *from the master* and *to the master*, i.e., $T^f = \{\Delta t_1, \Delta t_2, \Delta t_3\}$ (black values) and $T^t = \{\Delta t_1, \Delta t_2\}$ (blue values).

In case of bi-directional inter-arrival times monitoring, we compute Δt times of each subsequent packets exchanged between the master and slave, see Figure 5.2 (b), thus we get one distribution $T = \{\Delta t_1, \Delta t_2, \Delta t_3, \Delta t_4, \ldots\}$.

One-directional model better describes communication profile of a sending station while bi-directional model efficiently represents typical data exchange between a pair of stations. Which model is more suitable for statistical-based anomaly detection depends on behavior of underlying ICS protocols. Bi-directional distribution gives more sense for a master-slave communication while one-directional modeling better fits a publish-subscribe data exchange.

An important question for any statistical modeling is what statistical distribution describes the best the chosen packet features? In case of packet inter-arrival times, there are several studies observing their statistical distribution. Perkins showed in [97] that inter-arrival time of RTP packets is similar to Gaussian dis-

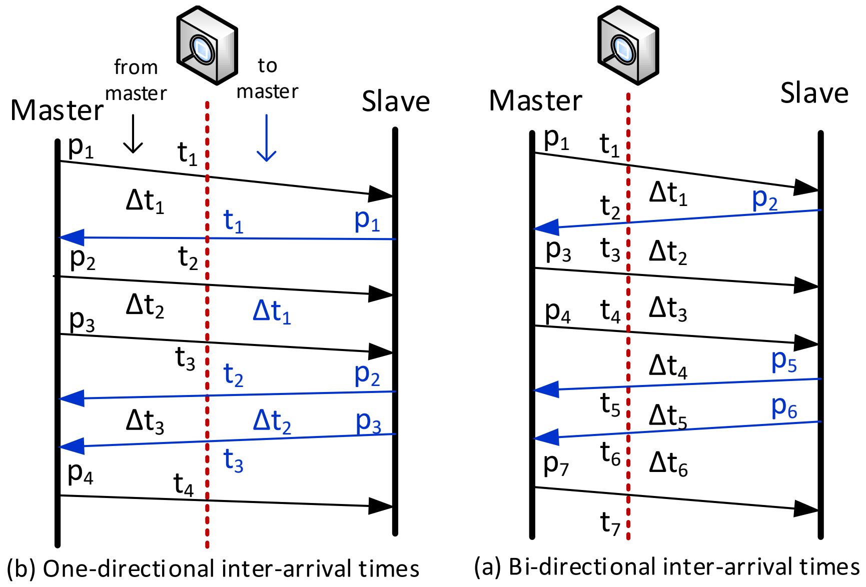


Figure 5.2: One-directional and bi-directional inter-arrival times

tribution. Santos da Silva et al. [109] also supposed normal distribution in their classification of SDN traffic using flow features. Lin et al. [79] approximated distribution of timing properties of ICS protocols (S7, Modbus, IEC 104) by normal distribution using the Central Limit Theorem. Also our experiments of measuring inter-arrival times of GOOSE and IEC 104 traffic proves that this feature can be approximated with normal distribution. Real inter-arrival times distribution can be a bit skewed due to the congestion, but congestion is rare in industrial networks.

5.2.4 Network Traffic Profiles

Industrial communication is usually limited to a fixed number of communicating nodes connected to the ICS network. In the *master-slave* communication model, a master station communicates with one or more slaves. This behavior is typical for industrial protocols like IEC 104, MMS, or Modbus. In the *publish-subscribe* communication model, the publisher sends regularly data to a set of receiving stations, mostly in one direction. This model is implemented by the GOOSE protocol, for instance. Based on the communication model, we define two kinds of network profiles that we use for statistical modeling of network communication: *master-oriented profile* and *peer-to-peer-oriented profile*.

The master-oriented profile, see Figure 5.3, describes a communication where a master periodically exchanges data with a set of slaves. Typically, slaves are requested at the master one by one using round-robin order. To build the profile, we select all network transmissions identified by master's ID, e.g., IP or MAC address. Then the statistical model for the master-oriented profile repre-

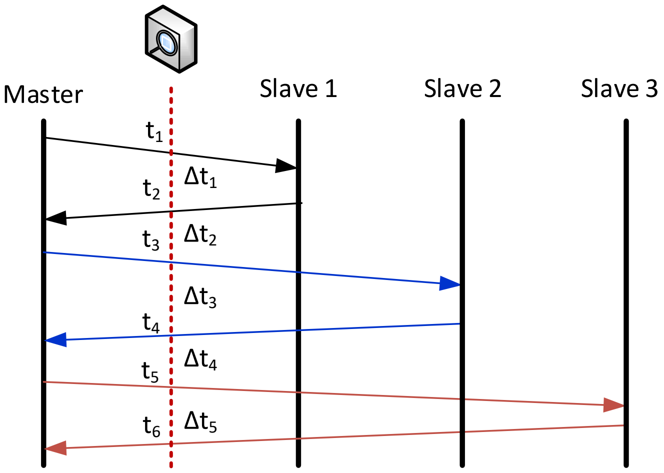


Figure 5.3: Master-oriented communication profile

sents inter-arrival times of packets sent or received by this master node as a set $T = \{\Delta t_1, \Delta t_2, \Delta t_3, \ldots\}$. Additionally, we can refine the model by adding packet direction and measure Δt times between subsequent packets of each direction as mentioned in previous section. Then we create two subsets based on direction, i.e., $T^f = \{\Delta t_1^f, \Delta t_2^f, \ldots\}$ for the *from-master direction* and $T^t = \{\Delta t_1^t, \Delta t_2^t, \ldots\}$ for the *to-master direction*.

A peer-to-peer-oriented profile represents communication between two nodes, e.g., IEC 104 master and slave, or GOOSE publisher and its subscribers, see Figure 5.4. In this case, peer-to-peer communication is identified by peer IDs, i.e, a pair

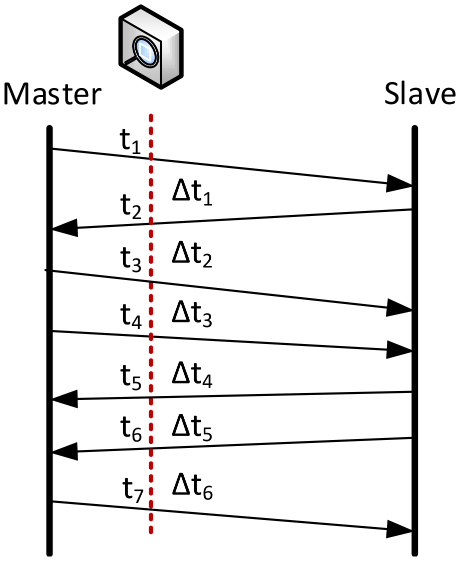
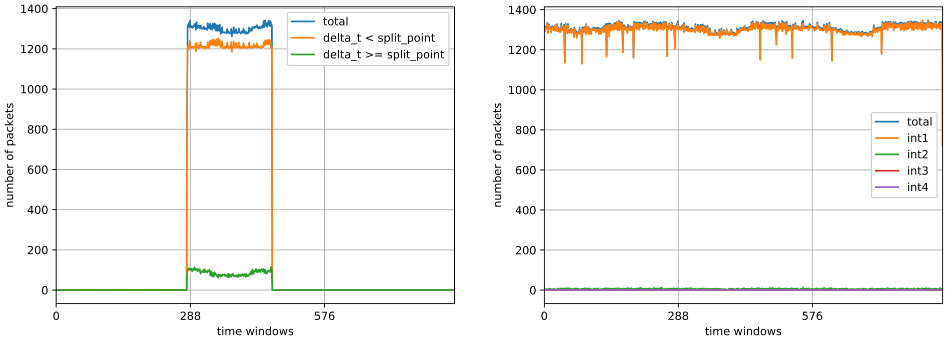


Figure 5.4: Peer-to-peer-oriented communication profile

of MAC addresses, IP addresses, and/or ports, or using a multicast destination address like in case of GOOSE communication. The peer-to-peer profile is built for each pair of communicating devices. The number of pairs is determined during the learning phase and usually stays unchanged until a new device is connected to the network which is quite rare in smart grid networks.

The difference between master-oriented profile and peer-to-peer oriented profile is highlighted in Figure 5.5. The figure displays the number of packets of in `13122018` dataset, see Section 5.3.1, transmitted in five minute windows. The dataset contains communication of 14 different devices. Figure 5.5 (a) shows application of peer-to-peer-oriented communication profile in *from master* direction. The profile considers communication of one pair device only, i.e., the master and one slave. Looking at the graph, we can notice that during 2 days and 23 hours long data capturing, these two devices communicated only shortly, therefore it is not possible to get one stable statistical model for their communication. One solution is to create two statistical models which would, however, prevent detection of many attacks. More suitable solution is to apply master-oriented profile which covers all communications of the given master and where the packet distribution with inter-arrival time characteristic is stable as showed on Figure 5.5 (b).



(a) Peer-to-peer profile (b) Master-oriented profile

Figure 5.5: Network profiles of inter-arrival times, dataset `13122018`

5.3 Statistical Modeling of ICS Traffic

Now we describe how to build a statistical model of ICS communication using two packet features: inter-arrival packet times and packet direction. Packets that are subject of modeling are selected using the previously defined network profiles. The core of our method is how to automatically find suitable split points that divide Δt times characteristic of a given communication into regions which provide more subtle models. As described more thoroughly below, split points represent an additional characteristic that creates more stable models for representing ICS communication.

The text of this section is structured as follows. First, we describe datasets used for our experiments and present three approaches how to automatically select split points. Then we show how the statistical model is computed using the Three Sigma Rule and selected split points.

5.3.1 Datasets

For our experiments we used several datasets with IEC 104 and GOOSE traffic listed in Table 5.1. The first four datasets were created at our university, datasets RTU8, RTU11, and RICS were provided by RTS Labs in Linköping University, Sweden. The last dataset with GOOSE communication was captured at GIGS Labs in Grenoble, France. Each line in the table contains the name of the dataset, number of captured packets, number of packets of interest, i.e, IEC 104 or GOOSE, duration of capturing and the number of communicating ICS devices.

	Packets			
Dataset	**Total**	**IEC/Goose**	Duration	Devices
13122018 (I)	1,433,083	874,697	2 days 23h	14
10122018 (I)	102.971	62,676	4h 53 min	4
14-12-18 (I)	35,905	14,342	15h 38min	2
17-12-18 (I)	150,273	58,929	2 days 20h	2
RTU8 (I)	5,788,789	3,117,663	6 days 18h	2
RTU11 (I)	3,491,020	1,828,733	6 days 18h	2
RICS (I)	4,477,807	882,957	12 days 21h	2
Goose (G)	200,583	83,966	19h 26 min	4

Table 5.1: Datasets with IEC 104 (I) and GOOSE (G) traffic.

5.3.2 Finding Split Points

As mentioned in the previous research [79, 109, 118], packet inter-arrival time Δt is a useful characteristic for describing network behavior. In this work, we create a statistical model of the number of transmitted packets having specific characteristic(s), e.g., Δt, packet size, direction, within a fixed time window. To build an accurate model, we first split observed packets into several regions based on their characteristic(s) Then we create a statistical model for each region. Usage of regions provides more accurate models which enable to detect common anomalies and identify a particular anomaly that occurred in the network. This particularly helps for ICS communication that exhibits periodic patterns. Instead of detecting periodic cycles, their size and duration as in [23], we use regions that provide sufficient accuracy of modeling of periodicity in communication with much less complexity. This is demonstrated on DoS attack in Section 5.4.2.

In addition, application of regions helps to identify of an anomaly type which is important especially for network administrators. This section presents three possible ways how to determine split points for inter-arrival times.

Four equal regions. The first case represents a naïve solution that splits the range of inter-arrival times into four equal regions based on the maximal and minimal values observed in the training dataset, i.e., the total range $R = \Delta t_{Max} - \Delta t_{Min}$ is split into four regions:

$$\langle \Delta t_{Min}, \frac{1}{4} \times R), \langle \frac{1}{4} \times R, \frac{1}{2} \times R), \langle \frac{1}{2} \times R, \frac{3}{4} \times R), \langle \frac{3}{4} \times R, \Delta t_{Max} \rangle$$

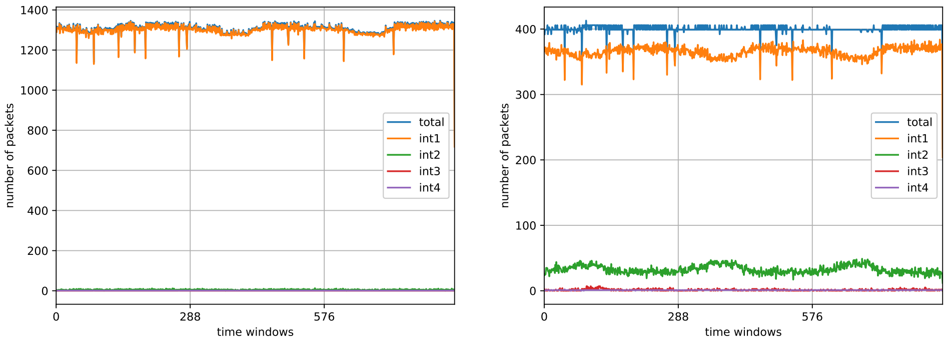
Unsurprisingly, this solution is not very suitable for ICS traffic modeling, since Δt values are usually not uniformly distributed as seen in Table 5.2. The table shows inter-arrival time distribution of packets in direction *from master* (fm) and *to master* (tm). Taking the first row with $\Delta t_{Min} = 0$ and $\Delta t_{Max} = 16.19$, we obtain four regions $\langle 0, 4.05)$, $\langle 4.05, 8.1)$, $\langle 8.1, 12.15)$, and $\langle 12.15, 16.2\rangle$. However, the majority of observed Δt values fall into the first region, since even the third quartile (75%) includes values with $\Delta t \leq 0.0004$. Typically, inter-arrival time of few packets is much greater then inter-arrival time of the rest of packets. With such splitting, the majority of packets falls into one or two regions.

Dataset	Direction	min	Q_1	Q_2	Q_3	max
13122018	fm	0.0000	0.0000	0.0003	0.0004	16.1905
	tm	0.0000	0.0002	0.0004	0.0600	10.1331
10122018	fm	0.0000	0.0000	0.0003	0.0005	8.2033
	tm	0.0000	0.0002	0.0004	0.0598	5.2006
14-12-18	fm	0.0000	1.6701	3.2010	5.2896	19.7166
	tm	0.0000	1.0076	3.0301	6.0784	19.2687
17-12-18	fm	0.0000	1.9989	3.5909	5.6002	19.9873
	tm	0.0001	1.0091	3.0332	6.0831	19.2696
RTU8	fm	0.0000	0.2025	0.2044	0.2184	1.2111
	tm	0.0000	0.0142	0.0145	0.0146	15.5452
RTU11	fm	0.0000	0.2109	0.3734	0.4792	2.4896
	tm	0.0000	0.0060	0.0121	0.0145	1.4055
RICS	fm	0.0000	0.0464	0.0830	3.8960	20.0577
	tm	0.0000	0.0073	0.0124	0.1410	10.1876

Table 5.2: Inter-arrival time distribution in selected datasets

Non-uniform distribution of inter-arrival times is apparent on graphs in Figure 5.6 which display the number of packets transmitted in five minute windows of `13122018` dataset in *from master* and *two master* directions.

The figure presents master-oriented profiles where additional characteristics show the effect of using four equally large intervals of inter-arrival times. The



(a) Direction *from master* (b) Direction *to master*

Figure 5.6: Split points selected equally, dataset *13122018*

blue line depicts the total number of packets, i.e., the sum of all regions. You can notice that the most packets fall into region *int1* (orange line) while the other three regions of equal size contain only few values. For *from master* direction, the average number of packets matching the first region *int1* is 1300.5 packets, for the second region *int2* it is 5.2 packets, for the third region *int3* 0 packet and for the last region *int4* 1 packet. For *to master* direction, the average number of packets matching the four regions is 365.1 (*int1*), 32.5 (*int2*), 1.2 (*int3*), and 1.0 (*int4*), see Table 5.3.

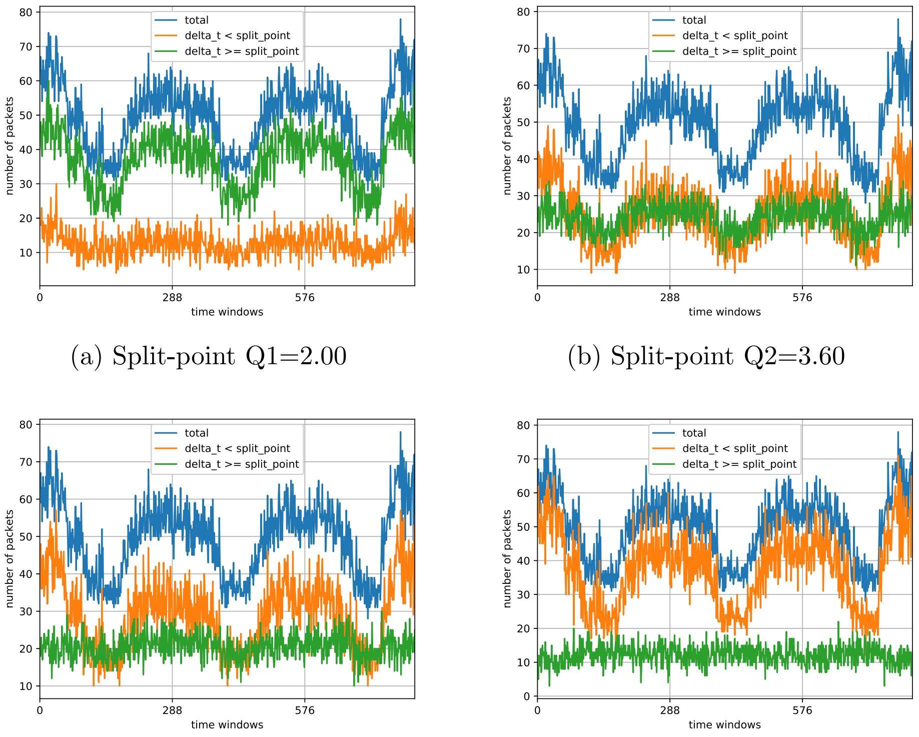
	Region 1	**Region 2**	**Region 3**	**Region 4**
fm regions	$\langle 0; 4.05)$	$\langle 4.05; 8.1)$	$\langle 8.1; 12.14)$	$\Delta t \geq 12.14$
packets	1300.5	5.2	0	1.0
tm regions	$\langle 0; 2.53)$	$\langle 2.53; 5.07)$	$\langle 5.07; 7.6)$	$\Delta t \geq 7.6.$
packets	365.1	32.5	1.2	1.0

Table 5.3: Equal regions for dataset `13122018` (a 5-minute window)

The pre-defined split points. Table 5.2 highlights significant differences in inter-arrival times distribution for individual datasets and directions. It is obvious that reasonable split-points cannot be defined ad-hoc. To define suitable split points, we need to analyze inter-arrival times of the given learning dataset. Due to this fact we decided to reduce the number of split points and Δt regions. We search for one split point for each direction only that provides two additional characteristics of the traffic expressed in regions.

With non-uniform inter-arrival time distribution it seems reasonable to choose some percentile, e.g. median, as a recommended split point. Such solution provides much better characteristics than the equal regions. As obvious in Table 5.2

it is not easy to find a single percentile which gives the best split point selection for each dataset and direction. The choice of a suitable quartile for one dataset and direction leads to less stable characteristics for another dataset or direction. For example, in dataset 17-12-18 and direction *from master* the 75th percentile Q3 produces a very stable characteristic while the opposite direction of the same percentile produces unstable periodic characteristics, see Figs. 5.7 (d) and 5.8 (d).



(a) Split-point Q1=2.00

(b) Split-point Q2=3.60

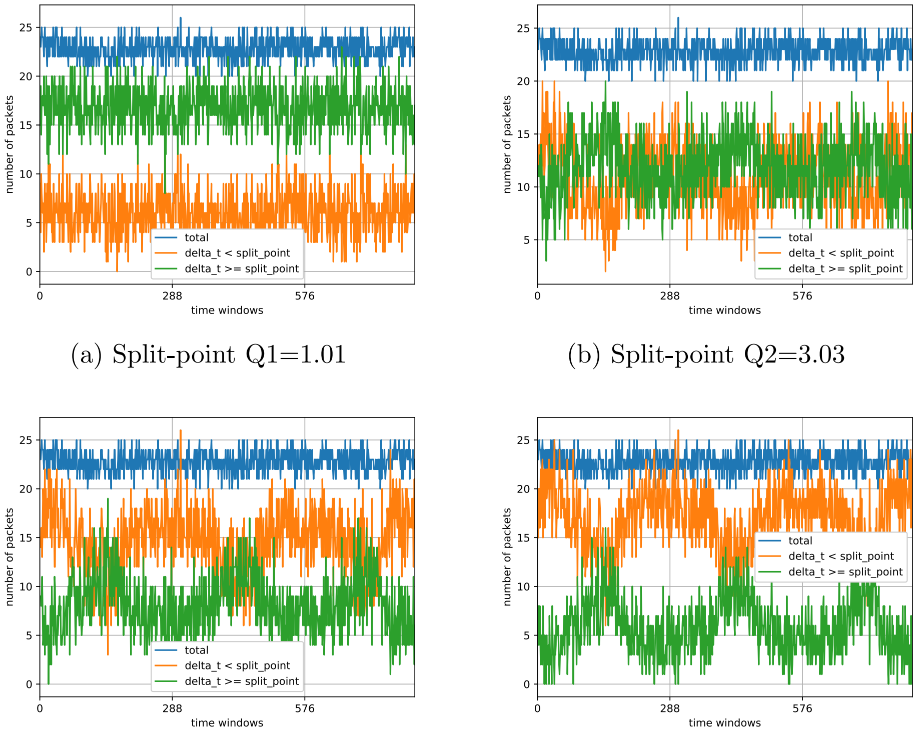
(c) Split-point mean=4.13

(d) Split-point Q3=5.66

Figure 5.7: Pre-defined split points, *from master* direction, dataset *17-12-18*

Figure 5.7 depicts the number packets of 17-12-18 dataset transmitted in five minute windows in *from master* direction using the peer-to-peer profile. The figure demonstrates selection of different split points, i.e, quartiles Q1, Q2, mean, and Q3. We can see that when using the mean and Q2 for splitting inter-arrival times into regions, it is not easy to describes these regions statistically because they overlaps. The overlapping would result in putting a packet into several regions and statistical profiles which would distort the model of communication. Thus, more suitable candidates for this datasets are Q1 and Q3.

Similarly, when observing graphs in Figure 5.8, suitable candidates for split points are also Q1 and Q3 with none or small overlapping. However, for other datasets, split point candidates may differ, see our experiments described in [24]. Thus, we can conclude that split points cannot be effectively selected by splitting



(a) Split-point Q1=1.01

(b) Split-point Q2=3.03

(c) Split-point mean=4.29

(d) Split-point Q3=6.08

Figure 5.8: Pre-defined split points, *to master* direction, dataset *17-12-18*

the inter-arrival range of the dataset into four equal regions neither by choosing predefined values representing quartiles and the mean. The third approach shows an automated way how to find suitable split points for a learning dataset by a two-step algorithm that includes computation of the mean and standard deviation for split point candidates and selection of the one that best fulfills matching conditions.

Automatically derived split points. The third case employs the automated method that finds suitable split points for individual directions of the given traffic. This approach utilizes distribution of Δt of packets transmitted in the given direction together with the standard deviation. Undoubtedly, different split points produce different characteristics. Some split points can filter-out the periodic behavior from at least one characteristic. This case is more suitable for anomaly detection as it produces more stable characteristics. Depending on traffic, the most suitable split points are the median of inter-arrival times, and quartiles Q1 or Q3.

Instead of testing the value of each percentile of inter-arrival times as a candidate split-point, we approximate the inter-arrival times distribution with four values: quartiles Q1, Q2, mean and Q3. We search for the best split-point among these four candidates.

Tables 5.4 and 5.5 show the candidate split-points for dataset `17-12-18` with *mean* and *standard deviation* (std) of the resulting characteristics. Δt distribution is derived from the first 48 hours of the captured traffic in order to decrease the influence of the periodicity. For *from master* direction, see Table 5.4, the value 5.66 (Q3) produces characteristic with the minimal standard deviation, so Q3 is the best candidate for the split point. For *to master* direction, see Table 5.5, we obtain the most stable characteristic for value 1.01, so Q1 is the best candidate.

Split point		**Δt < split_point**		**Δt ≥ split_point**	
candidate	**value**	**mean**	**std**	**mean**	**std**
Q1	2.00	12.66	3.80	36.82	8.08
Q2	3.60	25.29	8.34	24.19	3.94
mean	4.13	28.89	9.38	20.59	3.33
Q3	**5.66**	37.36	10.99	12.12	**2.83**

Table 5.4: Automated split point selection, dataset `17-12-18`, *from master*

Split-point		**Δt < split_point**		**Δt ≥ split_point**	
candidate	**value**	**mean**	**std**	**mean**	**std**
Q1	**1,01**	6,04	2,33	16,78	**2,23**
Q2	3,03	10,92	3,13	11,90	2,86
mean	4,29	14,61	3,34	8,21	3,01
Q3	6,08	16,90	3,37	5,93	3,02

Table 5.5: Automated split point selection, dataset `17-12-18`, *to master*

Therefore, our algorithm for automated selection of split points tests four candidates for split points: Q1, Q2, mean and Q3 and select the candidate that produces the most stable characteristic, i.e., it has the minimal standard deviation. However, for some datasets the split point with minimal deviation divides packets in such way that the stable characteristic contains only few packets in each time window. If the mean is close to zero, then the testing the Three Sigma Rule range would go to negative values and such model would not be capable to detect some types of anomalies. For this reason, we put an additional condition on selection of the split point from the set of candidates: we search for the candidate (i) with the minimal standard deviation of the distribution, and (ii) matching the non-zero condition for the Three Sigma Rule, i.e., $mean - 3 \times \sigma > 0$, see Section 5.2.2.

5.3.3 Building the Statistical Profile

In this part, we provide a step-by-step description of the ICS modeling process. Consider an input sequence of packets in master-oriented or peer-to-peer-oriented communication model, i.e., between two devices or a master and the corresponding slaves as described in Section 5.2.4. For each packet i in the sequence, extract timestamp t_i and create a set $T^d = (t_1^d, t_2^d, ..., t_n^d)$, where $d \in \{t, f\}$ denotes direction *to master* (t) and *from master* (f). We build up the statistical model for the given set of packets as follows:

1. For each subsequent packets (i) an $(i+1)$ determine the packet inter-arrival time $\Delta t_{i+1}^d = t_{i+1}^d - t_i^d$ and create the sequence $\Delta T^d = (\Delta t_1^d, \Delta t_2^d, \ldots, \Delta t_n^d)$ describing inter-arrival times of the communication for given direction.

2. Partition sets T^d and ΔT^d into subsets based on direction, i.e., T^f and ΔT^f for *from master* direction and T^t and ΔT^t for *to master* direction.

3. For each set ΔT^d where $d \in \{f, t\}$, determine a set of split point candidates $D^d = \{Q1^d, Q2^d, mean^d, Q3^d\}$ where $Q1$ is the first quartile, $Q2$ is the median, $Q3$ is the third quartile and $mean$ is the arithmetic mean.

4. For each split point candidate $sp \in D^d$ split the ΔT^d into two sets $S^{d,L}$ and $S^{d,U}$, where $S^{d,L}$ is a set of $\Delta t_i^d < sp$ (lower region) and $\Delta t_i^u \geq sp$ (upper region). For each regions, find the number of packets transmitted in every time window and compute its mean and standard deviation using equations (5.1) and (5.2).

5. From the set of split point candidates select for each direction the best split point with minimum σ that satisfies condition $m - 3 \times \sigma > 0$.

6. Filter out the time windows in which the number of transmitted packets does not satisfy the Three Sigma Rule. Find the new mean m' and standard deviation σ' for the reduced set of time windows. This step is important for learning when Δt values of few data points are too far from majority of values. Such exceptions happen in ICS communication. By refining the mean and standard deviation in this step keeps the computed region small enough to match majority of normal data points. When omitting this refinement, the created region would become too wide which would cause the increase number of false negatives.

7. For a filtered set of time windows, create the statistical profile

   $$P^d = (sp^d, \langle a_1, a_2 \rangle, \langle l_1, l_2 \rangle, \langle u_1, u_2 \rangle)$$

   where sp is the selected split point from D^d, $a_1 = m' - 3 \times \sigma', a_2 = m' + 3 \times \sigma', l_1 = m'^l - 3 \times \sigma'^l, l_2 = m'^l + 3 \times \sigma'^l$, and $u_1 = m'^u - 3 \times \sigma'^u, u_2 = m'^u + 3 \times \sigma'^u$.

   Range $\langle a_1, a_2 \rangle$ denotes the total number of the transmitted packet in the time window, range $\langle l_1, l_2 \rangle$ denotes the packets with inter-arrival times less than the split point (lower region) and range $\langle u_1, u_2 \rangle$ denotes the packets with inter-arrival times greater than the split point (upper region).

By applying this method, we can create statistical profiles for our datasets using packet inter-arrival times and packet direction, see Table 5.6.

Dataset	Dir	Split point	Total range	Lower region	Upper region
13122018	fm	0.09	<1260.54;1357.95>	<1171.15;1270.40>	<45.17;131.89>
	tm	0.46	<390.15;411.42>	<336.42;359.35>	<46.27;59.52>
10122018	fm	0.10	<1200.78;1413.44 >	<1104.74;1311.42>	<79.51;119.30>
	tm	0.40	<367.10;427.92>	<314.46;375.33>	<47.91;54.49>
14-12-18	fm	5.28	<17.74;82.24>	<0.27;72.10>	<4.82;22.70>
	tm	1.01	<19.39;26.28>	<-1.43;12.03>	<11.09;23.98>
17-12-18	fm	5.66	<20.30;76.94>	<4.70;68.05>	<3.90;20.62>
	tm	1.01	<19.39;26.22>	<-1.01;12.26>	<10.67;23.64>
RTU8	fm	0.186	<1329.00;1858.74>	<86.53;524.54>	<1230.18;1346.55>
	tm	(0.014	<147.58;206.59>	<10.04;58.02>	<92.85;193.18>
RTU11	fm	0.211	<368.79;1302.63>	<-231.92;644.10>	<541.62;713.65>
	tm	0.006	<40.97;144.76>	<9.69;36.95>	<11.34;127.58>
RICS	fm	1.35	<169.52;248.56>	<114.27;191.58>	<50.29;62.05>
	tm	0.14	<24.56;33.61>	<14.25;29.47>	<0.84;13.56>

Table 5.6: Example of statistical profiles for our dataset

The following section presents how the statistical profiles of smart grid communication can be used for anomaly detection.

5.4 Anomaly Detection

Validation tests were performed to confirmed suitability of the proposed statistical method for ICS communication and correctness of defined ranges for individual features selected to describe behavior of the communication. For each test we used the first two thirds of the captured communication to learn the statistical profile using the steps described in the previous section. Then we validated the profile on the last third of the dataset.

5.4.1 Simple Detection and 3-Value Detection Methods

We employed two detection methods: simple detection and 3-values detection. The *simple detection* evaluates each time window independently and compares the number of transmitted packets of a selected communication (using the peer-to-peer or master-oriented communication profile) in this single window with the learned statistical profile. An anomaly is detected if a value does not fit the specified range of the profile determined by the Three Sigma Rule, see Section 5.2.2.

By doing experiments with the simple detection, we noticed that the method marks a number of five minute windows as anomalies even if it is a normal traffic, see the second column in Table 5.7.

Such anomalies expressed as exceeding inter-arrival times mostly disappear in the consecutive time windows. These small deviations from the normal range of values can be caused by communication overhead of a device or network delays.

	Simple detection		3-value detection	
Dataset	FP/all	Accuracy	FP/all	Accuracy
13122018	2/285	99.30%	0/285	100%
10122018	1/20	95%	0/20	100%
14-12-18	0/63	100%	0/63	100%
17-12-18	4/273	98.53%	0/273	100%
RTU8	9/650	98.62%	5/650	99.23%
RTU11	16/650	97.54%	0/650	100%
RICS	37/1240	97.02%	11/1240	99.11%
Goose	4/78	94.87%	0/78	100%

Table 5.7: Validation tests summary

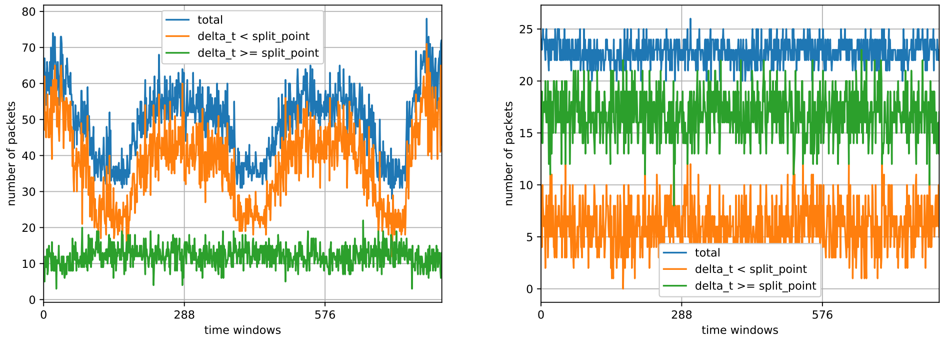
By close examination we found out that such anomaly windows did not appear in bursts but were scattered over the whole communication period, see detailed results in [24]. In addition, the characteristics returned back to the normal range in the subsequent time window. We identified such behavior in all our datasets for both directions (from master and to master). This means that such short deviations representing the legitimate traffic appear once in a while in communication and should not be treated as false positives (FP).

There are several ways how to eliminate these false positives. The easiest solution is to enlarge the normal values range so that the exceeding values match the profile. However, this approach would increase the number of false negatives, i.e., the number of cases where an illegitimate traffic would be considered as normal and thus potential threats would not be detected. Another solution is to apply a sliding window that observes the immediate context of the given time window and smooths these small deviations. We implemented the approach in our second method called *3-value detection.*

The 3-value detection method evaluates three consecutive time windows. If two of three values fit the given profile range, the test is considered valid and no anomaly is announced. An anomaly is reported only if at least two of the three windows detect the values outside the specified range for some characteristics of the traffic. We show in the following section that this approximation has no negative effect on detection of common security threats. As seen in Table 5.7, 3-value detection produces better accuracy with less false positives.

5.4.2 Anomaly Detection Using Statistical Profiles

For anomaly detection, we made experiments with 3-value detection only. For these tests, we created a normal statistical profile using the same dataset as in case of probabilistic anomaly detection described in Section 4.5.2. Then we applied the profiles on the dataset with anomalies. The dataset with anomalies was manually created using typical attack vectors described in research papers[3]. Graphical representation profiles of *17-12-18* dataset is depicted in Figure 5.9.



(a) Normal traffic profile (*from master*) (b) Normal traffic profile (*to master*)

Figure 5.9: Statistical profiles of dataset *17-12-18*

DoS attack and rogue device. Table 5.8 lists the five minute windows that were revealed as anomalies. The header lists the real five-minute windows in which the attack occurred. We used to datasets. The first one included two DoS attacks in time windows 110-128 and 142-161. The second dataset comprised a rogue device that connected to the network and in time windows 8-13 communicated instead of a legitimate device. In the table we can see windows in which the attack was detected by individual characteristics. Arrows indicate whether the number of packets was above or below the range of specified values.

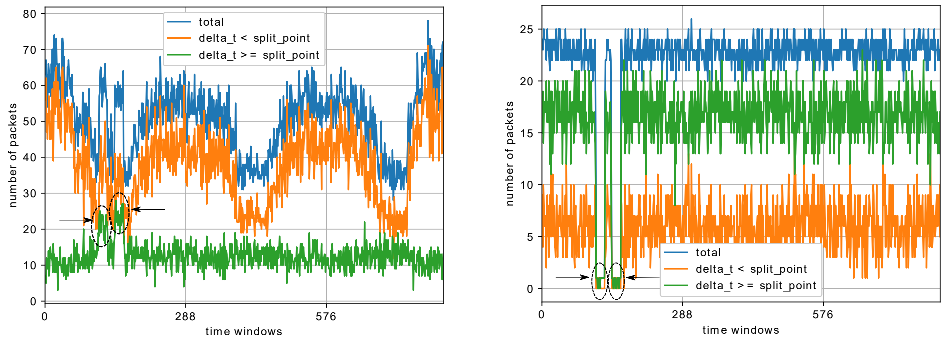
Direction	Profile	DoS Attack		Rogue Device
		110-128	142-161	8-13
fm	total	-	-	10-14
	$\Delta t < sp$	-	-	10-14
	$\Delta t \geq sp$	112-114, 117-121, 125-128 ↑	145-161 ↑	10-14
tm	total	111-130 ↓	143-162 ↓	9-14 ↓
	$\Delta t < sp$	-	-	-
	$\Delta t \geq sp$	111-129 ↓	143-162 ↓	9-14 ↓

Table 5.8: DoS attack and rogue device detection

The table shows that two DoS attacks that occurred during time windows 110-128 and 142-161 were successfully detected. Both DoS attacks were found in the upper region ($\Delta t \geq sp$) characteristic of the *from master* (fm) direction and in both the total and upper region characteristics of the *to master* (tm) direction. In

[3] Available as CSV traces at https://github.com/matousp/datasets [May 2021].

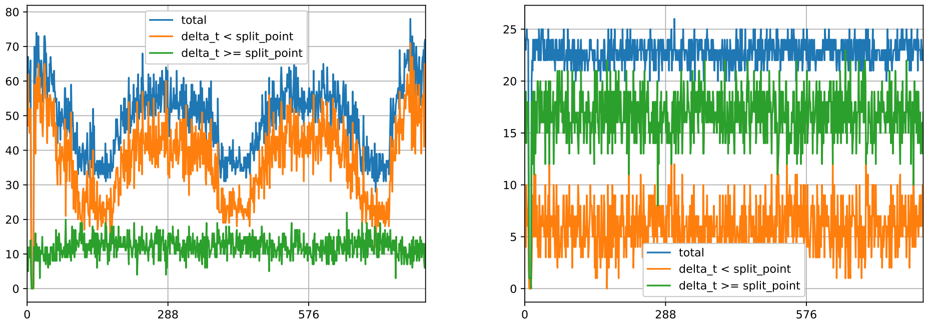
the *from master* direction, the expected number of packets with the given characteristics was higher then the normal value, in the opposite direction it was smaller. This indicates that the DoS attack was directed against the slave, see Figure 5.10.



(a) DoS attack (*from master*) (b) DoS attack (*to master*)

Figure 5.10: DoS attack detection

The presence of a rogue device connected to the ICS network was found in both directions, see also Figure 5.11. Here, we can notice a fast drop of packets in windows 10-14 caused by packets replaced by a rogue device. The drop is seen in the lower and upper regions in the *from master* direction and also in the upper region of the *to master* direction.



(a) A rogue device (*from master*) (b) A rogue device (*to master*)

Figure 5.11: Detection of a rogue device in the network.

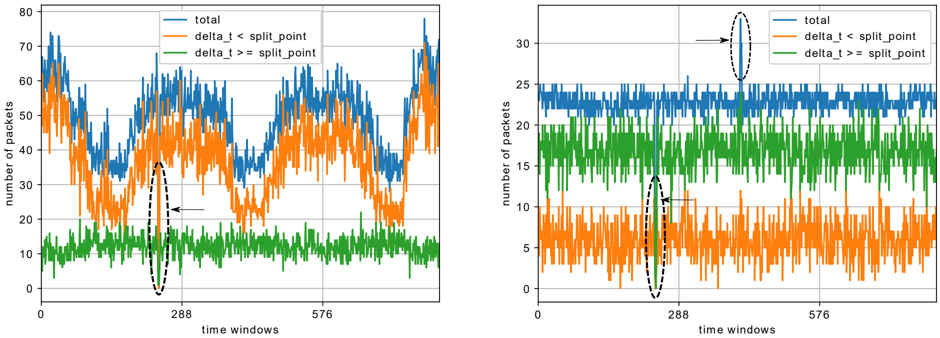
The experiment with DoS attack clearly justifies our approach of using additional characteristics for statistical anomaly detection. When using a simple statistics represented by the number of transmitted packets only (i.e., the total characteristics in our statistical profile), the DoS attack would not be revealed in the *from master* direction because both DoS attacks happened in the down part of the periodic behavior of the communication, see Figure 5.10, and excessive packets caused by the attack would fall into the expected total range approximated by the Three Sigma Rule.

However, our statistical profile is composed of three characteristics, i.e., the total number of transmitted packets and two regions created by the split point. Thus, we receive a small-grained model of the passing traffic which reveals details normally hidden in the total transmission statistics.

Scanning and Switching Attacks. Table 5.9 shows results of scanning and switching attack detection. Two scanning attacks appeared in time windows 239-242 and 413-417. The switching attack happened in time windows 190-192. The first scanning attack was detected in both directions, i.e., in the *from master* and *to master* directions, see Figure 5.12. The second scanning attack was detected only in the *to master* direction.

Direction	**Profile**	**Scanning attack**		**Switching attack**
		239-242	**413-417**	**190-192**
fm	total	240-242 ↓	-	-
	$\Delta t < sp$	241-242 ↓	-	-
	$\Delta t \geq sp$	240-242 ↓	-	-
tm	total	240-243 ↓	414-417 ↑	191-192 ↑
	$\Delta t < sp$	-	-	191-192 ↑
	$\Delta t \geq sp$	240-243 ↓	-	-

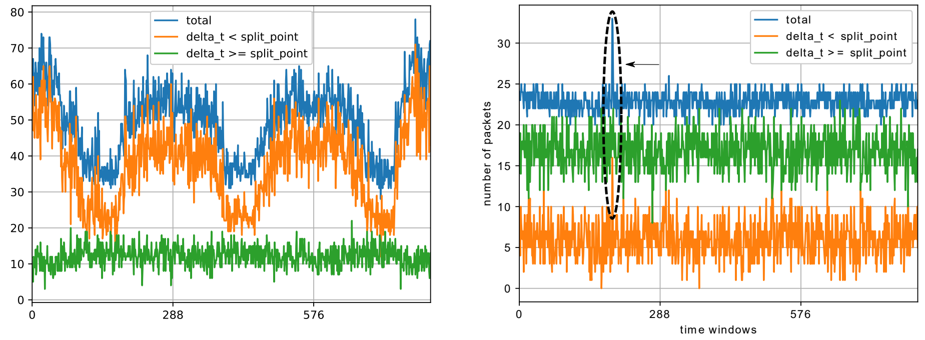
Table 5.9: Scanning and switching attack detection



(a) Scanning attack (*from master*) (b) Scanning attack (*to master*)

Figure 5.12: Scanning attacks detection

On the example we can demonstrate benefit of building profiles for each directions separately. This approach allows to detect some attacks that would be hidden in the overall traffic when using the bi-directional profile. The same is true also for the switching attack, see the right column in Table 5.9 and Figure 5.13, where the attack is not visible in the *from master* direction (it is too short) but is detected in the *to master* direction.



(a) Switching attack (*from master*)

(b) Switching attack (*to master*)

Figure 5.13: Switching attack detection

Connection loss and injection attack. The last experiment shows anomaly detection when connection is lost and during the injection attack, see Table 5.10. The scenario include connection loss in two time periods: windows 310-312 and 498-510. The injection attack appeared in two time windows 59-60 and 356-368.

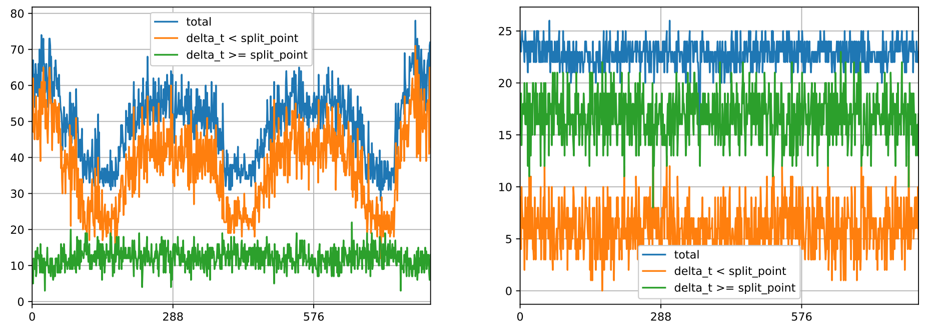
Direction	Profile	Connection loss		Injection attack	
		310-312	**498-510**	**59-60**	**365-368**
fm	total	311-312 ⇂	499-511 ⇂	-	-
	$\Delta t < sp$	-	500-510 ⇂	-	-
	$\Delta t \geq sp$	-	499-510 ⇂	-	-
tm	total	311-313 ⇂	499-511 ⇂	-	367-369 ⇂
	$\Delta t < sp$	-	-	-	-
	$\Delta t \geq sp$	311-312 ⇂	499-511 ⇂	-	-

Table 5.10: Connection loss and injection attack detection.

As seen in the table, connection loss attacks were properly detected but in case of the injection attacks, only the second attack was properly detected in the *to master* direction. The first injection attack was ignored because it did not involve a sufficient number of packets that would cause a significant deviation from the learned profile, see Figure 5.14.

5.4.3 Discussion

The result of our experiments proves that the proposed method based on observing statistical characteristics of inter-arrival times of transmitted packets is able to



(a) Injection attack (*from master*) (b) Injection attack (*to master*)

Figure 5.14: Injection attack detection

successfully detect more frequent anomalies that appear in smart grid networks. Statistical modeling of inter-arrival times distribution was fine-grained by splitting distribution to two regions based on the automatically selected splitting point in order to create more accurate profile of the traffic. Additionally, we create profiles for both directions individually which helps to improve detection of some attacks and overcome limits of periodicity that would absorb some attacks when using a simple statistics.

Table 5.11 summarizes capability of the proposed method to detect the individual attacks and compares it with our previous research using probabilistic automata [87]. The table shows that while the DoS attack is not properly detected by probabilistic automata, it can be detected using statistical profiles. On the contrary, statistical approach cannot detect an injection attack which is nevertheless covered by the probabilistic approach.

Anomaly	**Statistical-based AD**	**Automata-based AD**		
		Single	$Distr_{pref}$	$Distr_{aler}$
Connection loss	✓	×	✓	✓
Injection attack	✓/×	✓	✓	✓
DoS attack	✓	×	×	×
Rogue Device	✓	✓	✓	✓
Scanning attack	✓	✓	✓	✓
Switching attack	✓	✓	✓	✓

Table 5.11: Comparison of the anomaly detection based on statistical profiles and probabilistic automata.

Another advantage of building statistical profiles for each direction and the usage of split points for fine-grained representation of the normal traffic is that the method also identifies the type of an attack as depicted in Figure 5.15.

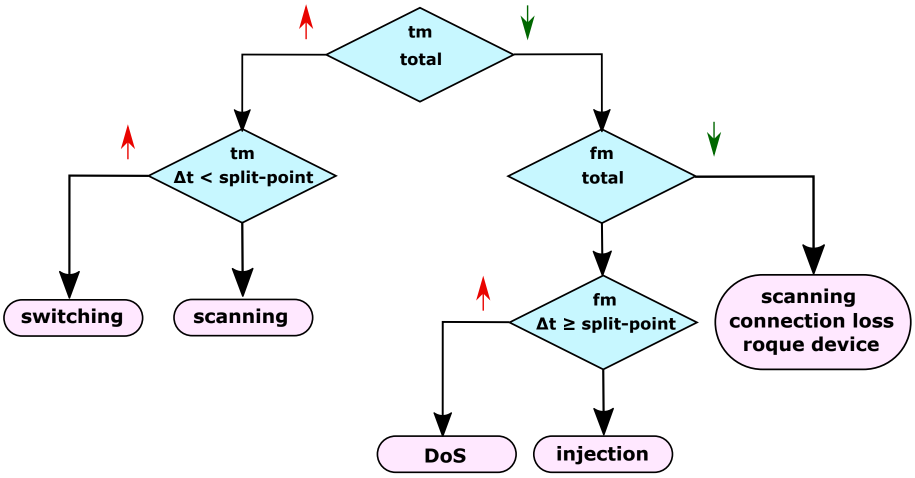


Figure 5.15: Attack Identification based on the change from expected values.

As the figure shows the reasoning is based on individual characteristics and combination of changes from the expected behavior. Such patterns help to identify an attack class. For example, a DoS attack is marked with the decreased number of packets in the total characteristic in the *to master* direction and the increased number of packets of upper region (i.e., $\Delta t \geq sp$) in the *from master* direction.

Similarly, the switching attack is usually marked by the increased number of packets of the total packet characteristic and also by the increased number of packets in the lower region (i.e., $\Delta < sp$). You can also notice that some attacks have various patterns (e.g., the scanning attack) that can overlap with other types of attacks. Nevertheless, even such indication of possible attack type is useful for network operators.

5.5 Summary

In this chapter we introduced a method for statistical modeling of inter-arrival times that employs several characteristics (total number, lower region, upper region, packet direction) in order to create a simple but accurate profile for smart grid communication. The profile is computed in two-steps process using the mean and standard deviation and outlier values tested by the Three Sigma Rule.

In the previous text we presented several approaches for selecting best split points to receive fine-grained characteristics of the traffic. The important aspect of anomaly detection method is to reduce the number of false positives. We have identified that false positives can be suppressed by the 3-value detection that compares three consecutive windows of monitored traffic to the model and raises an alert only if an anomaly is detected at least in two of them. We demonstrated that profiles built for each direction of ICS communication could detected the considered attacks.

The conducted experiments demonstrated high accuracy of the proposed method similar to more complex anomaly detection methods while the computation costs of building a profile and evaluating the monitored traffic are substantially lower. The method was demonstrated on a variety of datasets consisting of IEC 104 and Goose traffic.

We also compared the method with the previously proposed approach based on probabilistic automata. It is obvious from Table 5.11 that both method complement each other in case of DoS and injection attack and similarly gives good results for other types of anomalies. This result is important for deployment in production anomaly detection system where the result of one method can be proved or augmented by the result of the other method in order to give accuracy to the detection and decrease the number of false positives.

Chapter 6

Conclusion

This thesis presents the research of cyber security of industrial systems that was done by the author of the thesis during years 2016-2021. The research was focused on monitoring and detection of cyber attacks in smart grid control protocols which have been frequent targets of sophisticated cyber attacks in the past. Such attacks usually have serious consequences as presented in Section 2.4. In order to mitigate and eliminate attacks against control communication of industrial networks, we need (1) to increase visibility of communication, and (2) to apply detection methods for revealing attack traces in communication.

In this work we presented ICS flow monitoring concept based on extension of IPFIX standard used in IT networks. The proposed extension includes definition of ICS flow properties obtained from ICS protocols that are mapped into IPFIX records. Another extension represents virtual flows that encapsulate communication of Layer 2 industrial protocols into IPv6 link-local flows, or that divide multiple application-layer data units into separate flows for more detailed monitoring. The advantage of the proposed technique is that it uses the standardized IPFIX framework that can be easily incorporated into common network monitoring and management systems.

This work also presented two methods for anomaly detection of industrial protocols. The first method is based on probabilistic automata. We introduced new representations of ICS communication sequences using deterministic probabilistic automata and prefix trees. We showed how these models are automatically created from samples of normal ICS traffic. Then we presented two methods for anomaly detection based on computing probability of an observed sequence (single conversation reasoning) and on automata comparison (distribution reasoning). Our experiments show that both methods are able to detect common cyber attack vectors that are typical for industrial networks. In addition, they exhibit very high accuracy with a small number of false positives.

The proposed method is not limited to specific protocols or environment and can be adopted to any industrial protocol as presented in Section 4.3. In our future work we plan to apply the method on other industrial protocols and evaluate its precision. We also focus on its optimization, namely on reduction of states. Another open issue is modeling of corrupted communication where some packets are lost due to unstable communication or buffer overflow.

The second presented technique was statistical-based anomaly detection. This technique applies a different view on networking data. Instead of observing legitimate communication sequences, their order and occurrence, we monitor packet time properties of ICS communication. In our case, we model distribution of packet inter-arrival times of ICS transmissions. Since ICS communication is stable with periodic patterns, we can easily create statistical profiles of peer-to-peer or master-to-slaves communications. We showed that splitting packets into regions based on inter-arrival times and directions provide sufficient model representing normal communication. Using the model we can detect common anomalies and also identify their type.

Both proposed techniques can be combined in order to increase accuracy of detection and minimizing the number of false positives. They can be implemented in an anomaly detection module that works with IPFIX flow records or incorporated into industrial intrusion detection systems.

This work was presented to the research community at the 6th Conference on the Engineering of Computer Based Systems (ECBS 2019) [89], 6th International Symposium for ICS & SCADA Cyber Security Research in 2019 (ICS-CSR 2019) [88], the 7th Conference on the Engineering of Computer Based Systems in 2021 (ECBS 2021) [107], the 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM 2021) [87], and the 17th International Conference on Network and Service Management (CNSM 2021)[25]. Automata-based technique was published in the Journal of Information Security and Application in 2020 (JISA) [90].

Acronyms and Abbreviations

A-XDR Adjusted External Data Representation.

AARE Application Association Response.

AARQ Application Association Request.

ACSE Association Control Service Element.

ADS Anomaly Detection System.

AMI Advanced Metering Infractructure.

AMR Advanced Metering Reading.

APCI Application Protocol Control Information.

APDU Application Data Protocol Unit.

APT Advanced Persistant Threat.

ASDU Application Service Data Unit.

ASN.1 Abstract Syntax Notation One.

AWL Application Whitelisting.

BE BlackEnergy.

BER Basic Encoding Rules.

CI&A Confidentiality, Integrity, Availability.

DCS Distributed Control System.

DFFA Deterministic Frequency Finite Automaton.

DNP3 Distributed Network Protocol, Version 3.

DoS Denial of Service.

DPA Deterministic Probabilistic Automaton.

DPI Deep Packet Inspection.

GOOSE Generic Object Oriented Substation Event.

GSE Generic Substation Event.

HAN Home Area Network.

HMI Human Machine Interface.

ICS Industrial Control System.

IDS Intrusion Detection System.

IEC International Electrotechnical Commission.

IED Intelligent Electronic Device.

IETF Internet Engineering Task Force.

IoT Internet of Things.

IPFIX IP Flow Information Export.

ISO International Organization for Standardization.

LAN Local Area Network.

MMS Manufacturing Message Specification.

OT Operational Technology.

PCS Process Control System.

PPDU Presentation Protocol Data Unit.

PT Prefix Tree.

RTU Remote Terminal Unit.

SAS Substation Automation System.

SCADA Supervisory Control and Data Acquisition.

SIS Safety Instrumented System.

SMV Sampled Measured Values.

SPDU Session Protocol Data Unit.

TPDU Transport Protocol Data Unit.

VMD Virtual Manufacturing Device.

WAN Wide Area Network.

Bibliography

[1] Distribution automation using distribution line carrier systems - Part 4: Data communication protocols - Section 41: Application protocol - Distribution line message specification. Standard IEC 61334-4-41:1996, International Electrotechnical Commission, Geneva, August 1996.

[2] COSEM. Glossary of Terms. Technical Report DLSM UA 1002:2003, DLMS User Association, 2003.

[3] Industrial automation systems – Manufacturing Message Specification – Part 2: Protocol specification. Standard ISO 9506-2:2003, International Organization for Standardization, Geneva, June 2003.

[4] Distribution automation using distribution line carrier system - Part 6: A-XDR encoding rule. Technical Report IEC 61334-6:2000, International Electrotechnical Commission, 3, rue de Varembas Geneva, Switzerland, 2006.

[5] Electricity metering - Data exchange for meter reading, tariff and load control - Part 62: Interface classes. Technical Report IEC TS 62056-62:2006, International Electrotechnical Commission, 3, rue de Varembas Geneva, Switzerland, 2006.

[6] Blue Book: COSEM. Identification System and Interface Classes. Technical Report DLSM UA 1000-1:2010, DLMS User Association, 2010.

[7] Communication networks and systems for power utility automation - Part 7-2: Basic information and communication structure - Abstract communication service interface (ACSI). Standard IEC 61850-7-2:2010, International Electrotechnical Commission, Geneva, August 2010.

[8] Communication networks and systems for power utility automation - Part 8-1: Specific communication service mapping (SCSM) - Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3. Standard IEC 61850-8-1:2011, International Electrotechnical Commission, Geneva, June 2011.

[9] BlackEnergy & Quedagh. The convergence of crimeware and APT attacks. Technical report, F-Secure Labs, 2015.

[10] BlackEnergy. Technical article, New Jersey Cybersecurity & Communications Integration Cell, 2 Schwarzkopf Dr., Ewing Township, NJ 08628, July 2016.

[11] Electricity metering data exchange - The DLMS/COSEM suite - Part 5-3: DLMS/COSEM application layer. Standard IEC 62056-5-3:2017, International Electrotechnical Commission, Geneva, October 2017.

[12] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. A Survey of Network Anomaly Detection Techniques. *J. Netw. Comput. Appl.*, 60(C):19–31, January 2016.

[13] Ahmed Altaher, Stéphane Mocanu, and Jean-Marc Thiriet. Evaluation of Time-Critical Communications for IEC 61850-Substation Network Architecture. *CoRR*, abs/1512.07004, 2015.

[14] Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987.

[15] George Argyros, Ioannis Stais, Aggelos Kiayias, and Angelos D. Keromytis. Back in Black: Towards Formal, Black Box Analysis of Sanitizers and Filters. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 91–109, 2016.

[16] Michael J. Assante and Robert M. Lee. The Industrial Control System Cyber Kill Chain. Technical report, SANS Institute, October 2015.

[17] Michael J. Assante, Robert M. Lee, and Tim Conway. Modular ICS Malware. Technical report, Electricity Information Sharing and Analysis Center (E-ISAC), August 2017.

[18] R. R. R. Barbosa, R. Sadre, and A. Pras. A first look into SCADA network traffic. In *2012 IEEE Network Operations and Management Symposium*, pages 518–521, April 2012.

[19] R. R. R. Barbosa, R. Sadre, and A. Pras. Towards periodicity based anomaly detection in SCADA networks. In *Proceedings of IEEE 17th International Conference on Emerging Technologies Factory Automation (ETFA)*, pages 1–4, Sept 2012.

[20] Rafael R. R. Barbosa, Ramin Sadre, and Aiko Pras. Difficulties in Modeling SCADA Traffic: A Comparative Analysis. In *The 13th International Conference on Passive and Active Measurement*, pages 126–135, 2012.

[21] Rafael Ramos Regis Barbosa. *Anomaly detection in SCADA systems: a network based approach.* PhD thesis, University of Twente, 4 2014.

[22] Rafael Ramos Regis Barbosa, Ramin Sadre, and Aiko Pras. Flow whitelisting in SCADA networks. *International Journal of Critical Infrastructure Protection*, 6(3):150 – 158, 2013.

[23] R.R.R. Barbosa, R. Sadre, and Aiko Pras. Exploiting traffic periodicity in industrial control networks. *International journal of critical infrastructure protection*, 13:52–62, June 2016. eemcs-eprint-26932.

[24] Ivana Burgetová and Petr Matoušek. Statistical Methods for Anomaly Detection in Industrial Communication. Technical Report IT-TR-2021-01, Brno University of Technology, 2021.

[25] Ivana Burgetová, Petr Matoušek, and Ondřej Ryšavý. Anomaly Detection of ICS Communication Using Statistical Models. In *Proceedings of the 17th International Conference on Network Service Management (CNSM 2021)*, page 7, 2021.

[26] João Caberera, B. Ravichandran, and Raman Mehra. Statistical traffic modeling for network intrusion detection. pages 466 – 473, 02 2000.

[27] Marco Caselli, Emmanuele Zambon, and Frank Kargl. Sequence-aware Intrusion Detection in Industrial Control Systems. In *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, CPSS '15, pages 13–24, New York, NY, USA, 2015. ACM.

[28] Marco Caselli, Emmanuele Zambon, Jonathan Petit, and Frank Kargl. Modeling message sequences for intrusion detection in industrial control systems. In *Critical Infrastructure Protection IX*, pages 49–71, Cham, 2015.

[29] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly Detection: A Survey. *ACM Comput. Surv.*, 41(3), July 2009.

[30] Anton Cherepanov. Win32/Industroyer. A new threat for industrial control systems. Technical report, ESET, June 2017.

[31] B. Claise. *Cisco Systems NetFlow Services Export Version 9*. IETF RFC 3954, October 2004.

[32] B. Claise. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information*. IETF RFC 5101, January 2008.

[33] B. Claise and B. Trammel. *Information Model for IP Flow Information Export (IPFIX)*. IETF RFC 7012, September 2013.

[34] B. Claise, B. Trammel, and P. Aitken. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. RFC 7011, IETF, September 2013.

[35] Gordon Clarke and Deon Reynders. *Practical Modern SCADA Protocols. DNP3, IEC 60870.5 and Related Systems.* Newnes, 2008.

[36] Smart Grid Cybersecurity Committee. Guidelines for Smart Grid Cybersecurity. Technical Report NISTIR-7628r1, National Institute of Standards and Technology, 2014.

[37] Manuel Crotti, Maurizio Dusi, Francesco Gringoli, and Luca Salgarelli. Traffic Classification Through Simple Statistical Fingerprinting. *SIGCOMM Comput. Commun. Rev.*, 37(1):5–16, January 2007.

[38] Michael K. Daly. The Advanced Persistent Threat. In *Proceedings of the Usenix*, November 2009.

[39] Colin de la Higuera. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, New York, NY, USA, 2010.

[40] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[41] Dragos. CrashOverride. Analysis of the Threat of Electric Grid Operations. Technical report, Dragos Inc., June 2017.

[42] Samuel Drews and Loris D'Antoni. Learning symbolic automata. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACACS 2017)*, Lecture Notes in Computer Science, pages 173–189. Springer, Berlin, Heidelber, 2017.

[43] Olivier Dubuisson and Philippe Fouquart. *ASN.1: Communication Between Heterogeneous Systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[44] ENISA. Communication network dependencies for ICS/SCADA Systems. Technical report, European Union Agency for Network and Information Security (ENISA), December 2016.

[45] Gilberto Fernandes, J. Rodrigues, L. F. Carvalho, J. Al-Muhtadi, and M. L. Proença. A comprehensive survey on network anomaly detection. *Telecommunication Systems*, 70:447–489, 2019.

[46] David Formby, Anwar Walid, and Raheem Beyah. A case study in power substation network dynamics. 1(1), June 2017.

[47] S. Fuloria, R. Anderson, F. Alvarez, and K. McGrath. Key management for substations: Symmetric keys, public keys or no keys? In *2011 IEEE/PES Power Systems Conference and Exposition*, pages 1–6, March 2011.

[48] R. Gerhards. *The Syslog Protocol.* IETF RFC 5424, March 2009.

[49] Niv Goldenberg and Avishai Wool. Accurate modeling of modbus/tcp for intrusion detection in scada systems. *International Journal of Critical Infrastructure Protection*, 6(2):63 – 75, 2013.

[50] Andy Greenberg. Crash override: The malware that took down a power grid. *Electric Power Systems Research*, 2017.

[51] David Hanes, Gonzalo Salqueiro, Patrick Grossetete, Rob Barton, and Jereme Henry. *IoT Fundamentals. Networking Technologies, Protocol and Use Cases for the Internet of Things.* Cisco Press, 2017.

[52] Vojtěch Havlena, Lukáš Holík, and Petr Matoušek. Learning Probabilistic Automata in the Context of IEC 104. Technical Report IT-TR-2020-01, Brno University of Technology, 2020.

[53] Kevin E. Hemsley and Dr. Ronald E. Fisher. History of Industrial Control System Cyber Incidents. (INL/CON-18-44411-Revision-2), 12 2018.

[54] R. Hinden and S. Deering. *IP Version 6 Addressing Architecture.* IETF RFC 2373, July 1998.

[55] Maarten Hoeve. OT Security Monitoring. Use Cases: Detection Strategy. Technical report, European Network for Cyber Security, 2017.

[56] R. Hofstede, V. Bartoš, A. Sperotto, and A. Pras. Towards real-time intrusion detection for NetFlow and IPFIX. In *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*, pages 227–234, Oct 2013.

[57] Michael Horkan. Challenges for IDS/IPS Deployment in Industrial Control Systems. Technical report, SANS Institute, July 2015.

[58] Mans Hulden. Treba: Efficient Numerically Stable EM for PFA. In Jeffrey Heinz, Colin Higuera, and Tim Oates, editors, *Proceedings of the Eleventh International Conference on Grammatical Inference*, volume 21 of *Proceedings of Machine Learning Research*, pages 249–253, University of Maryland, College Park, MD, USA, 05–08 Sep 2012. PMLR.

[59] IEC. Telecontrol equipment and systems - Part 5-104: Transmission protocols - Network access for IEC 60870-5-101 using standard transport profiles. Standard IEC 60870-5-104:2006, International Electrotechnical Commission, Geneva, June 2006.

[60] Power systems management and associated information exchange - Data and communications security - Part 1: Communication network and system security - Introduction to security issues. Standard, International Electrotechnical Commission, Geneva, May 2007.

[61] Power systems management and associated information exchange - Data and communications security. Standard, International Electrotechnical Commission, Geneva, May 2018.

[62] Malte Isberner, Falk Howar, and Bernhard Steffen. Inferring Automata with State-Local Alphabet Abstractions. In Guillaume Brat, Neha Rungta, and Arnaud Venet, editors, *NASA Formal Methods*, pages 124–138, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[63] ITU-T. *Data Networks and Open System Communications: Connection-oriented Protocol for the Association Control Service Element: Protocol Specification*. X.227, 1995.

[64] ITU-T. *Information technology - Open Systems Interconnection - Connection-oriented Presentation Protocol: Protocol Specification*. X.226, November 1995.

[65] ITU-T. *Information technology - Open Systems Interconnection - Connection-oriented Session Protocol: Protocol Specification*. X.225, November 1995.

[66] ITU-T. *Information technology - Open Systems Interconnection - Protocol for providing the connection-mode transport service*. X.224, November 1995.

[67] Jacek Jarmakiewicz, Krzysztof Parobczak, and Krzysztof Maślanka. Cybersecurity protection for power grid control infrastructures. *International Journal of Critical Infrastructure Protection*, 18:20 – 33, 2017.

[68] Dong Jin, D. M. Nicol, and Guanhua Yan. An event buffer flooding attack in DNP3 controlled SCADA systems. pages 2614–2626, Dec 2011.

[69] S. S. Jung, D. Formby, C. Day, and R. Beyah. A first look at machine-to-machine power grid network traffic. In *IEEE International Conference on Smart Grid Communications*, pages 884–889, Nov 2014.

[70] T. Khalifa, K. Naik, and A. Nayak. A survey of communication protocols for automatic meter reading applications. *IEEE Communications Surveys Tutorials*, 13(2):168–182, Second 2011.

[71] Amit Kleinmann and Avishai Wool. A statechart-based anomaly detection model for multi-threaded SCADA systems. In *Int. Conference on Critical Information Infrastructures Security*, pages 132–144, 2015.

[72] Eric D. Knapp and Joel Thomas Langill. *Industrial Network Security. Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems*. Syngress, 2015.

[73] Oualid Koucham, Stéphane Mocanu, Guillaume Hiet, Jean-Marc Thiriet, and Frédéric Majorczyk. Efficient Mining of Temporal Safety Properties for Intrusion Detection in Industrial Control Systems. *IFAC-PapersOnLine*, 51(24):1043 – 1050, 2018. 10th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS 2018.

[74] Tammo Krueger, Hugo Gascon, Nicole Krämer, and Konrad Rieck. Learning Stateful Models for Network Honeypots. In *Proceedings of the 5th ACM Workshop on Security and Artificial Intelligence*, AISec '12, pages 37–48, New York, NY, USA, 2012. ACM.

[75] Yoojin Kwon, Sangyoum Lee, Ralph King, Jong In Lim, and Huy Kang Kim. Behavior analysis and anomaly detection for a digital substation on cyber-physical system. *Electronics (Switzerland)*, 8(3), 3 2019.

[76] Robert M. Lee, Michael J. Assante, and Tim Conway. Analysis of the Cyber Attack on the Ukrainian Power Grid. Defense Use Case. Technical report, Electricity Information Sharing and Analysis Center (E-ISAC), March 2016.

[77] G. Liang, J. Zhao, F. Luo, S. R. Weller, and Z. Y. Dong. A Review of False Data Injection Attacks Against Modern Power Systems. *IEEE Transactions on Smart Grid*, 8(4):1630–1638, July 2017.

[78] Chih-Yuan Lin and Simin Nadjm-Tehrani. Understanding IEC-60870-5-104 Traffic Patterns in SCADA Networks. In *The 4th ACM Workshop on Cyber-Physical System Security*, CPSS '18, pages 51–60, 2018.

[79] Chih-Yuan Lin, Simin Nadjm-Tehrani, and Mikael Asplund. Timing-based anomaly detection in SCADA networks. In *International Conference on Critical Information Infrastructures Security*, pages 48–59. Springer, 2017.

[80] Yao Liu, Peng Ning, and Michael K. Reiter. False data injection attacks against state estimation in electric power grids. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 21–32, New York, NY, USA, 2009. ACM.

[81] Z. Lu, X. Lu, W. Wang, and C. Wang. Review and evaluation of security threats on the communication networks in the smart grid. In *Milcom 2010 Military Communication Conference*, pages 1830–1835, Oct 2010.

[82] Z. Lu, W. Wang, and C. Wang. Modeling, Evaluation and Detection of Jamming Attacks in Time-Critical Wireless Applications. *IEEE Transactions on Mobile Computing*, 13(8):1746–1759, Aug 2014.

[83] Fabio Martinelli, Francesco Mercaldo, and Antonella Santone. Real-Time SCADA Attack Detection by Means of Formal Methods. In *28th Int. Conf. on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 231–236, 2019.

[84] Petr Matoušek. Analysis of DLMS Protocol. Technical Report FIT-TR-2017-13, Brno University of Technology, 2017.

[85] Petr Matoušek. Description and analysis of IEC 104 Protocol. Technical Report FIT-TR-2017-12, Brno University of Technology, 2017.

[86] Petr Matoušek. Description of IEC 61850 Communication. Technical Report FIT-TR-2018-01, Brno University of Technology, 2018.

[87] Petr Matoušek, Vojtěch Havlena, and Lukáš Holík. Efficient Modelling of ICS Communication For Anomaly Detection Using Probabilistic Automata. In *IFIP/IEEE International Symposium on Integrated Network Management*, pages 1–9, 2021.

[88] Petr Matoušek, Ondřej Ryšavý, and Matěj Grégr. Increasing Visibility of IEC 104 Communication in the Smart Grid. In *The 6th International Symposium for ICS & SCADA Cyber Security Research 2019*, pages 21–30. BCS Learning and Development Ltd, 2019.

[89] Petr Matoušek, Ondřej Ryšavý, and Matěj Grégr. Security Monitoring of IoT Communication Using Flows. In *Proceedings of the 6th Conference on the Engineering of Computer Based Systems*, ECBS '19, pages 1–9. Association for Computing Machinery, 2019.

[90] Petr Matoušek, Ondřej Ryšavý, Matěj Grégr, and Vojtěch Havlena. Flow based monitoring of ICS communication in the smart grid. *Journal of Information Security and Applications*, 54:102535, 2020.

[91] Peter Maynard, Kieran McLaughlin, and Berthold Haberler. Towards Understanding Man-In-The-Middle Attacks on IEC 60870-5-104 SCADA Networks. In *Proceedings of the 2Nd International Symposium on ICS & SCADA Cyber Security Research 2014*, ICS-CSR 2014, pages 30–42, UK, 2014. BCS.

[92] James McCarthy, Michael Powell, Keith Stouffer, CheeYee Tang, Timothy Zimmerman, William Barker, Titilayo Ogunyale, Devin Wynne, and Johnathan Wiltberger. Securing Manufacturing Industrial Control Systems: Behavior Anomaly Detection. Technical Report NISTIR-8219, National Institute of Standards and Technology, 2018.

[93] Bill Miller and Dale C. Rowe. A survey of SCADA and critical infrastructure incidents. In *In Proceedings of the 1st Annual conference on Research in information technology, RIIT '12*, pages 51–56. ACM, 2012.

[94] D. C. Montgomery and G. C. Runger. *Applied Statistics and Probability for Engineers*. John Wiley and Sons, 7th edition, 2018.

[95] Naiara Moreira, Elías Molina, Jesús Lázaro, Eduardo Jacob, and Armando Astarloa. Cyber-security in substation automation systems. *Renewable and Sustainable Energy Reviews*, 54:1552 – 1562, 2016.

[96] Jacqueline O'Leary, Josiah Kimble, Kelli Vanderlee, and Nalani Fraser. Insights into Iranian Cyber Espionage: APT33 Targets Aerospace and Energy Sectors and has Ties to Destructive Malware, 2017.

[97] Colin Perkins. *RTP: Audio and Video for the Internet*. Addison-Wesley, 2003.

[98] U. K. Premaratne, J. Samarabandu, T. S. Sidhu, R. Beresh, and J. Tan. An Intrusion Detection System for IEC61850 Automated Substations. *IEEE Transactions on Power Delivery*, 25(4):2376–2383, Oct 2010.

[99] R. Presuhn, J.Case, K.McCloghrie, M.Rose, and S.Waldbusser. *Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)*. IETF RFC 3416, December 2002.

[100] André Proto, Leandro A. Alexandre, Maira L. Batista, Isabela L. Oliveira, and Adriano M. Cansian. *Statistical Model Applied to NetFlow for Network Intrusion Detection*, pages 179–191. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[101] Friedrich Pukelsheim. The Three Sigma Rule. *The American Statistician*, 48(2):88–91, 1994.

[102] Slavica V. Boštjančič Rakas, Mirjana D. Stojanović, and Jasna D. Marković-Petrović. A review of research work on network-based scada intrusion detection systems. *IEEE Access*, 8:93083–93108, 2020.

[103] R.Hinden and S.Deering. *IP Version 6 Addressing Architecture*. IETF RFC 4291, February 2006.

[104] Marshall T. Rose and Dwight E. Cass. *ISO Transport Service on top of the TCP Version: 3* . IETF RFC 1006, May 1987.

[105] Alec Russell. CIA plot led to huge blast in Siberian gas pipeline. *The Telegraph, UK*, February 2004.

[106] Mary-Ann Russon. US fuel pipeline hackers 'didn't mean to create problems'. *BBC News*, May 2021.

[107] Ondřej Ryšavý and Petr Matoušek. A Network Traffic Processing Library for ICS Anomaly Detection. In *ECBS '21: Proceedings of the 7th Conference on the Engineering of Computer Based Systems*, pages 144–151. Association for Computing Machinery, 2021.

[108] Omar Santos. *Network Security with NetFlow and IPFIX. Big Data Analytics for Information Security.* Cisco Press, 2016.

[109] Anderson Santos Da Silva, Cristian Cleder Machado, Rodolfo Vebber Bisol, Lisandro Zambenedetti Granville, and Alberto Schaeffer-Filho. Identification and Selection of Flow Features for Accurate Traffic Classification in SDN. In *IEEE 14th Int. Symposium on Network Computing and Applications*, pages 134–141, 2015.

[110] Roman Schlegel, Sebastian Obermeier, and Johannes Schneider. A security evaluation of IEC 62351. *Journal of Information Security and Applications*, 34:197 – 204, 2017.

[111] Klaus Schwab. *The Fourth Industrial Revolution*. Crown Publishing Group, USA, 2017.

[112] Z. Shelby, K. Hartke, and C. Bromann. *The Constrained Application Protocol (CoAP).* IETF RFC 7252, June 2014.

[113] Keith Stouffer, Victoria Pillitteri, Marshall Abrams, and Adam Hahn. Guide to Industrial Control Systems (ICS) Security. Technical Report NIST-SP-800-82r2, National Institute of Standards and Technology, 2015.

[114] M. Strobel, N. Wiedermann, and C. Eckert. Novel weaknesses in iec 62351 protected smart grid control systems. In *2016 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 266–270, Nov 2016.

[115] Chih-Che Sun, Adam Hahn, and Chen-Ching Liu. Cyber security of a power grid: State-of-the-art. *International Journal of Electrical Power & Energy Systems*, 99:45 – 56, 2018.

[116] John W. Tukey. *Exploratory data analysis*. Addison-Wesley, 1977.

[117] A. Valdes and S. Cheung. Communication pattern anomaly detection in process control systems. In *2009 IEEE Conference on Technologies for Homeland Security*, pages 22–29, May 2009.

[118] Pal Varga. Analyzing Packet Interarrival Times Distribution to Detect Network Bottleneck. In *IFIP EUNICE: Networks and Applications Towards a Ubiquitously Connected World*, volume 196, pages 134–141, 2006.

[119] Enrique Vidal, Franck Thollard, Colin de la Higuera, Francisco Casacuberta, and Rafael C. Carrasco. Probabilistic finite-state machines-part i. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(7):1013–1025, July 2005.

[120] Ladislav Šťastný, Lešek Franek, and Petr Fiedler. Wireless communications in smart metering. *IFAC Proceedings Volumes*, 46(28):330 – 335, 2013. 12th IFAC Conference on Programmable Devices and Embedded Systems.

[121] Cynthia Wagner, Jérôme François, Radu State, and Thomas Engel. Machine learning approach for ip-flow record anomaly detection. In Jordi Domingo-Pascual, Pietro Manzoni, Sergio Palazzo, Ana Pont, and Caterina Scoglio, editors, *NETWORKING 2011*, pages 28–39, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[122] Wenye Wang and Zhuo Lu. Cyber security in the smart grid: Survey and challenges. *Computer Networks*, 57(5):1344 – 1371, 2013.

[123] Yipeng Wang, Zhibin Zhang, Danfeng (Daphne) Yao, Buyun Qu, and Li Guo. Inferring protocol state machine from network traces: A probabilistic approach. In Javier Lopez and Gene Tsudik, editors, *Applied Cryptography and Network Security*, pages 1–18, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[124] L. Xie, Y. Mo, and B. Sinopoli. False data injection attacks in electricity markets. In *2010 First IEEE International Conference on Smart Grid Communications*, pages 226–231, Oct 2010.

[125] Y. Xu, Y. Yang, T. Li, J. Ju, and Q. Wang. Review on cyber vulnerabilities of communication protocols in industrial control systems. In *2017 IEEE Conference on Energy Internet and Energy System Integration (EI2)*, pages 1–6, Nov 2017.

[126] Y. Yan, Y. Qian, H. Sharif, and D. Tipper. A Survey on Smart Grid Communication Infrastructures: Motivations, Requirements and Challenges. *IEEE Communications Surveys Tutorials*, 15(1):5–20, First 2013.

[127] Y. Yuan, Z. Li, and K. Ren. Modeling load redistribution attacks in power systems. *IEEE Transactions on Smart Grid*, 2(2):382–390, June 2011.

Appendix A

ICS Protocols in the Smart Grid

Security monitoring of ICS communication deals with deep analysis of transmitted data and detection of unusual behavior based on gathered data. In this chapter we describe the most common ICS protocols that are present in Smart Grid networks. We will focus on typical operations, built-in security, and known vulnerabilities. The last section of the chapter introduces a unified model of Smart Grid communication that will be later used for security monitoring and incidents detection.

A.1 IEC 104

IEC 60870-5-104 protocol (aka IEC 104) is a part of IEC Telecontrol Equipment and Systems Standard IEC 60870-5 that provides a communication profile for sending basic telecontrol messages between two systems in electrical engineering and power system automation. Telecontrol means transmitting supervisory data and data acquisition requests for controlling power transmission grids.

IEC 104 provides the network access to IEC 60870-5-101 (aka IEC 101) using standard transport profiles. In simple terms, it delivers IEC 101 messages as application data (L7) over TCP, port 2404. IEC 104 enables communication between control station and a substation via a standard TCP/IP network. The communication is based on the client-server model. IEC 101/104 communication is exchanged between the controlled and the controlling station where *the controlled* station is monitored or commanded by a master station (RTU). The controlled station is also called outstation, remote station, RTU, 101-Slave, or 104-Server. *The controlling station* is a station where a control of outstations is performed (SCADA). Typically, it is a PC with SCADA system, can be also a RTU32.

IEC 101/104 defines several modes of direction:

- Monitor Direction is a direction of transmission from controlled station (RTU) to the controlling station (PC).

- Control Direction is a direction from controlling station, a SCADA system, to the controlled station, an RTU.

Protocol format

IEC 60870-5-104 is transmitted over TCP/IP protocol suite. It includes Application Protocol Control Information (APCI) layer and selection of Application Service Data Units (ASDUs) over it.

Each APCI (Application Protocol Control Information) starts with a start byte with value 0x68 followed by the 8-bit length of APDU (Application Protocol Data Unit) and four 8-bit control fields (CF). APDU contains an APCI or an APCI with ASDU, see Figure A.1. Generally, the length of APCI is 6 bytes. APCI is followed by an Application Service Data Unit (ASDU, also called telegram).

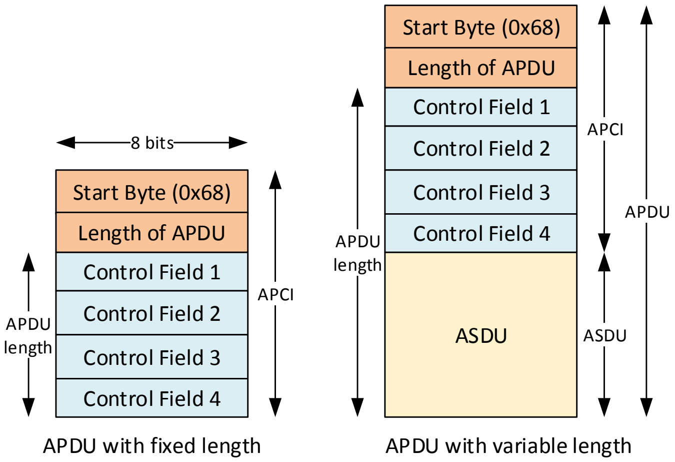


Figure A.1: APCI frame format

The ASDU contains two main sections: the data unit identifier (with the fixed length of six bytes), and the data itself, made up of one or more information objects. The data unit identifier defines the specific type of data, provides addressing to identify the specific identity of the data, and includes additional information as cause of transmission. Each ASDU can transmit maximum 127 objects. The format of ASDU is in Figure A.2.

Structure qualifier (SQ) specifies how information objects or elements are addressed. SQ=0 denotes a sequence of information objects which means addressing of individual single information elements or combination of information elements in a number of information objects (IO) of the same type. SQ=1 means addressing of just one information object, i.e., addressing of a sequence of single information elements or equal combinations of information elements of a single object per ASDU.

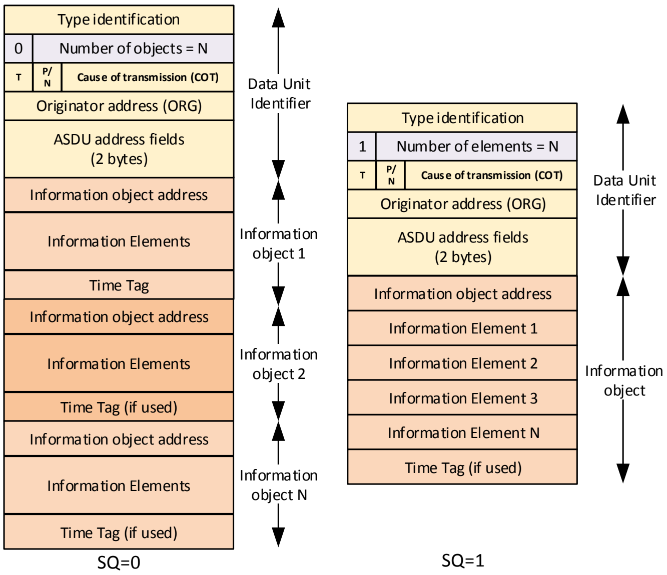


Figure A.2: The structure of ASDU with SQ=0 and SQ=1

Cause of Transmission (COT) field is used to control the routing of messages both on the communication network, and within a station, directing by ASDU to the correct program or task for processing. ASDUs in control direction are confirmed application services and may be mirrored in monitor direction with different causes of transmission. Each ASDU type has a defined subset of valid COT codes, see [85] for details.

ASDU Address Field (Common Address of ASDU, COA), is denotes a station address, however it can be structured to form a station/sector address where individual stations are broken up into multiple logical units.

ASDU transmits information objects within its structure. Each information object is addressed by Information Object Address (IOA) which identifies the particular data within a defined station. The address length is 3 bytes for IEC 104.

Basic Application Functions

Following application functions are implemented in IEC 101 communication:

- *Data acquisition* collects data cyclically, upon change, or upon request. In unbalanced transmission, the controlled outstation must always wait for a request from the controlling station. When balanced transmission is used, the buffered data is transmitted by the controlled outstation to the controlling station without a delay.

- *Event acquisition.* Events occur spontaneously at the application level of the controlled outstation. The transmission in balanced or unbalanced mode is similar to the data acquisition.

- *Interrogation* is used for updated the controlling station after an internal initialization. The controlling station requests the controlled outstations to transmit the actual values of all their process variables.

- *Clock synchronization.* After system initialization, the clocks are initially synchronized by the controlling station. After, the clocks are periodically resynchronized by transmission of a clock synchronization command.

- *Command transmission* is used to change the state of operational equipment. A command may be initiated by an operator or by automatic supervisory procedures in the controlling station.

- *Transmission of integrated totals* transmits values that are integrated over a specific time period using methods *Freeze-and-Read* for acquisition of integrated totals, and *Clear-and-Read* for acquisition of incremental information.

- *Changes in protocol and link parameters.*

- *Acquisition of transmission delay* which is needed for time correction.

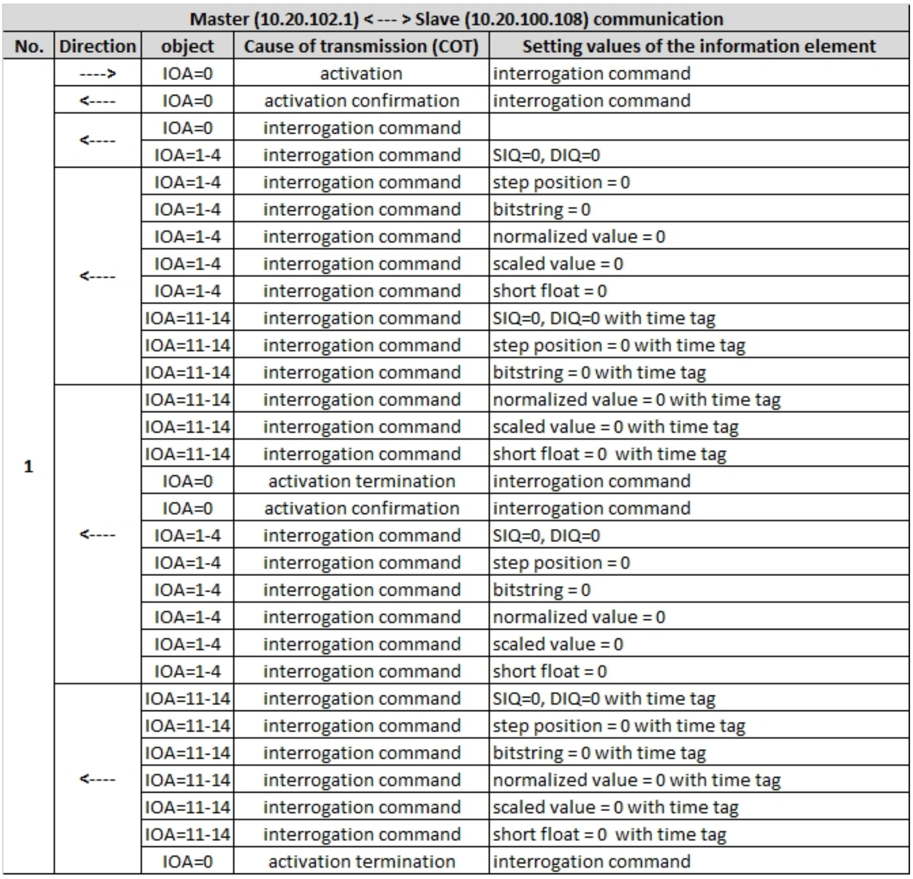
Communication Flow

Figure A.3 shows slave initialization which is sent as one logical transaction between the master and the slave. Individual IEC 104 packets contain several ASDU units. Each ASDU can transmit more information objects related to physical objects controlled by the slave.

Following Figure A.4 shows a set of logical transactions within one TCP stream. It demonstrates remote monitoring and control. It shows logical transactions exchanged between the master and slave. The arrow represents direction: master to slave (—>) or slave to master (<—). A transaction usually concerns one information object with its address (IOA). Only exception is transaction no. 1 which summarizes initialization of the system. Following transactions are initiated by the master station that sends activation command (COT=6) which is responded by the slave station using a sequence of messages with COT=7 (activation confirmation), COT=10 (activation termination) and COT=3 (spontaneous).

By analyzing typical IEC 104 communication, we can notice following features:

- Destination is addressed on L7 by common ASDU address (COA) which is 10 in this case (address of the slave station). Then, each destination contains several objects addressed by the information object address (IOA). Usually, the controlling station sends or retrieves data from the specific information object identified by its IOA.



Master (10.20.102.1) < --- > Slave (10.20.100.108) communication				
No.	**Direction**	**object**	**Cause of transmission (COT)**	**Setting values of the information element**
1	---->	IOA=0	activation	interrogation command
	<----	IOA=0	activation confirmation	interrogation command
	<----	IOA=0	interrogation command	
		IOA=1-4	interrogation command	SIQ=0, DIQ=0
	<----	IOA=1-4	interrogation command	step position = 0
		IOA=1-4	interrogation command	bitstring = 0
		IOA=1-4	interrogation command	normalized value = 0
		IOA=1-4	interrogation command	scaled value = 0
		IOA=1-4	interrogation command	short float = 0
		IOA=11-14	interrogation command	SIQ=0, DIQ=0 with time tag
		IOA=11-14	interrogation command	step position = 0 with time tag
		IOA=11-14	interrogation command	bitstring = 0 with time tag
	<----	IOA=11-14	interrogation command	normalized value = 0 with time tag
		IOA=11-14	interrogation command	scaled value = 0 with time tag
		IOA=11-14	interrogation command	short float = 0 with time tag
		IOA=0	activation termination	interrogation command
		IOA=0	activation confirmation	interrogation command
		IOA=1-4	interrogation command	SIQ=0, DIQ=0
		IOA=1-4	interrogation command	step position = 0
		IOA=1-4	interrogation command	bitstring = 0
		IOA=1-4	interrogation command	normalized value = 0
		IOA=1-4	interrogation command	scaled value = 0
		IOA=1-4	interrogation command	short float = 0
	<----	IOA=11-14	interrogation command	SIQ=0, DIQ=0 with time tag
		IOA=11-14	interrogation command	step position = 0 with time tag
		IOA=11-14	interrogation command	bitstring = 0 with time tag
		IOA=11-14	interrogation command	normalized value = 0 with time tag
		IOA=11-14	interrogation command	scaled value = 0 with time tag
		IOA=11-14	interrogation command	short float = 0 with time tag
		IOA=0	activation termination	interrogation command

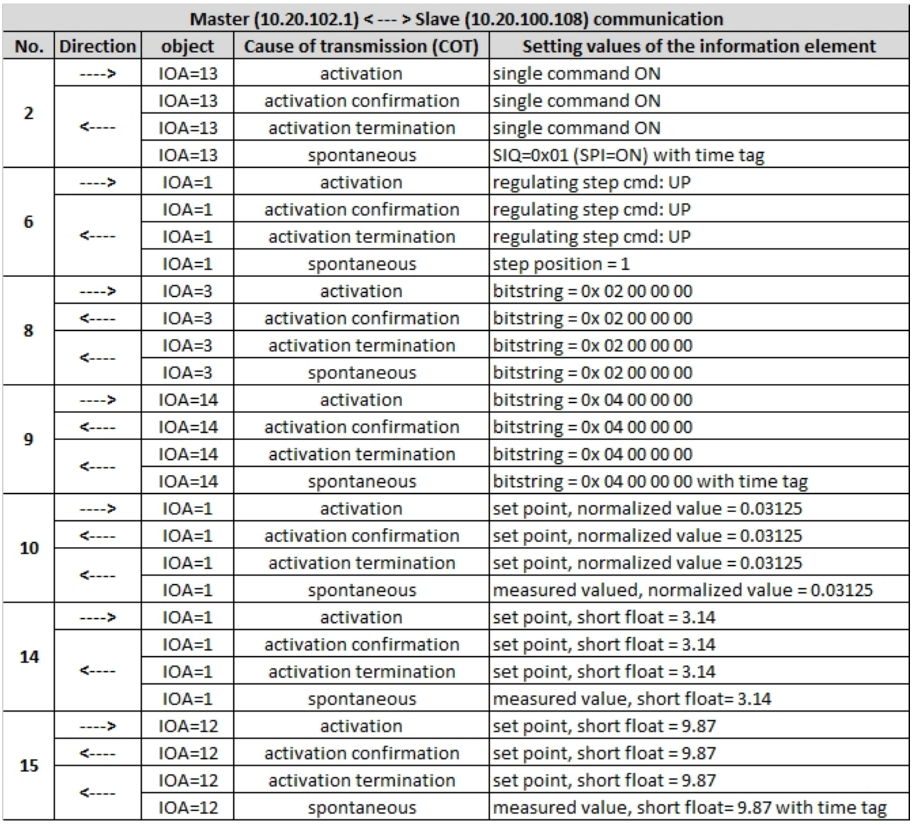
Figure A.3: IEC 104 slave initialization

- Special destination address 0 does not refer to a specific information object but to the configuration of the whole slave system. Thus, initialization of the slave is addressed using IOA=0.

- The transaction 1 sends an activation message with the interrogation command which enforces a sequence of interrogation answers from object 1-4 and 11-14 transmitting the actual settings of their values.

- We can notice that one object with a given IOA can contain several types of information elements. E.g., object 1 has elements of type SIQ, DIG, step position, bitstring, normalized value, scaled value, or short floating point.

- Some responses can be divided into several TCP packets, e.g., the response on transaction no. 4 is sent in two packets: one contains ActCon ASDU, the other ActTerm ASDU and Spon ASDU. This is different to transaction no. 3, where the response is sent via one TCP packet that contains three ASDUs: ActCon, ActTerm and Spon.



Master (10.20.102.1) < --- > Slave (10.20.100.108) communication				
No.	**Direction**	**object**	**Cause of transmission (COT)**	**Setting values of the information element**
2	---->	IOA=13	activation	single command ON
	<----	IOA=13	activation confirmation	single command ON
		IOA=13	activation termination	single command ON
		IOA=13	spontaneous	SIQ=0x01 (SPI=ON) with time tag
6	---->	IOA=1	activation	regulating step cmd: UP
	<----	IOA=1	activation confirmation	regulating step cmd: UP
		IOA=1	activation termination	regulating step cmd: UP
		IOA=1	spontaneous	step position = 1
8	---->	IOA=3	activation	bitstring = 0x 02 00 00 00
	<----	IOA=3	activation confirmation	bitstring = 0x 02 00 00 00
	<----	IOA=3	activation termination	bitstring = 0x 02 00 00 00
		IOA=3	spontaneous	bitstring = 0x 02 00 00 00
9	---->	IOA=14	activation	bitstring = 0x 04 00 00 00
	<----	IOA=14	activation confirmation	bitstring = 0x 04 00 00 00
	<----	IOA=14	activation termination	bitstring = 0x 04 00 00 00
		IOA=14	spontaneous	bitstring = 0x 04 00 00 00 with time tag
10	---->	IOA=1	activation	set point, normalized value = 0.03125
	<----	IOA=1	activation confirmation	set point, normalized value = 0.03125
	<----	IOA=1	activation termination	set point, normalized value = 0.03125
		IOA=1	spontaneous	measured valued, normalized value = 0.03125
14	---->	IOA=1	activation	set point, short float = 3.14
	<----	IOA=1	activation confirmation	set point, short float = 3.14
		IOA=1	activation termination	set point, short float = 3.14
		IOA=1	spontaneous	measured value, short float= 3.14
15	---->	IOA=12	activation	set point, short float = 9.87
	<----	IOA=12	activation confirmation	set point, short float = 9.87
	<----	IOA=12	activation termination	set point, short float = 9.87
		IOA=12	spontaneous	measured value, short float= 9.87 with time tag

Figure A.4: IEC 104 slave initialization

- Some objects (e.g., 11-14) returns values with timestamp, others (1-4) do not use them.
- One TCP packet can transmit a sequence of ASDUs of several objects. E.g, the third response from slave in the transaction 1 transmits ASDUs with objects IOA=1-4 and IOA=11-14.
- We can also see that some ASDUs contain one information element and some a sequence of information elements. This number is specified in the Number of Objects field in the ASDU header.

A.2 IEC 61850 MMS

MMS (Manufacturing Message Specification) is a messaging system for modeling real devices and functions and for exchanging information about the real device, and exchanging process data – under real-time conditions – and supervisory control information between networked devices and/or computer applications.

MMS is defined by standards ISO/IEC 9506-1 (Services) and ISO/IEC 9506-2 (Protocol).

- The service specification contains definition of the Virtual Manufacturing Device (VMD), services (and messages) exchanged between nodes on a network, and the attributes and parameters associated with the VMD and services.
- The protocol specification defines the rules of communication, i.e., the sequencing of messages across the network, the format and encoding of the messages, and the interaction of the MMS layer with the other layers of the communications network. MMS communicates using a client-server model. A client is a network application or device (e.g., monitoring system, control center) that asks for data or an action from the server. A server is a device or application that contains a Virtual Manufacturing Device (VMD) and its objects (e.g., variables) that the MMS client can access. The VMD object represents a container in which all other objects are located, see Figure A.5. The client issues MMS service requests and the server responds to these requests.

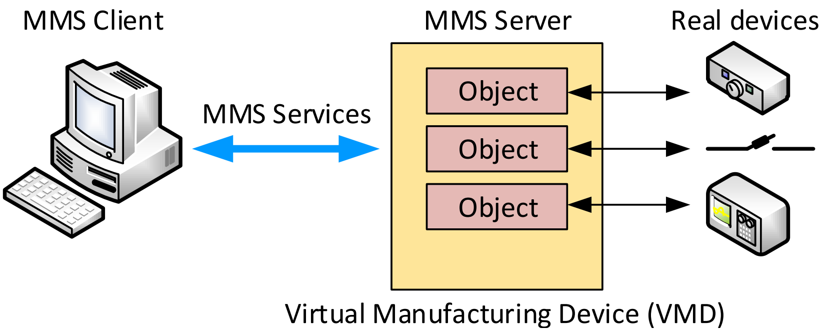


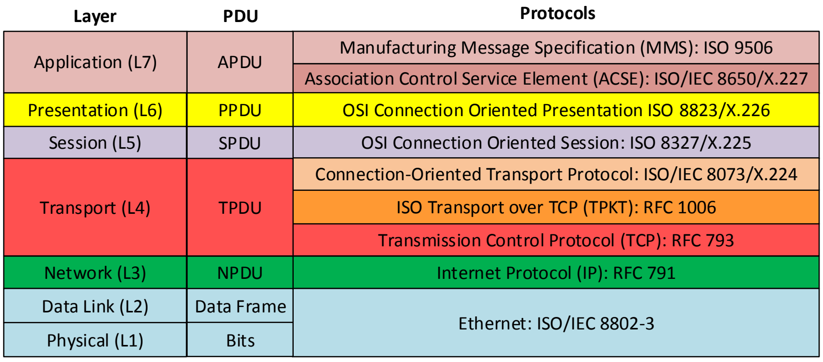
Figure A.5: MMS client-server model

MMS uses an object-oriented approach with object classes (Named Variable, Domain, Program invocation), instances and methods (read, write, store, start, stop, etc.).

MMS Communication

MMS does not specify how to address clients and servers and relies on the addressing scheme of underlying protocols. In practice, clients and servers are addresses by their IP address and the MMS is encapsulated over TCP, port 102. However, port 102 is dedicated to ISO TSAP Class 0 which is general encapsulation of ISO model protocols over TCP. Upper layers use ISO identifiers, e.g., TSAP (transport service access point), COTP source and destination references, OSI calling and called session selectors, etc.

The encapsulation includes several ISO protocols which are part of ISO stack, see Figure A.6. Not all the protocols shall be presented in every MMS message. Detailed description of each layer can be found in [86].



Layer	PDU	Protocols
Application (L7)	APDU	Manufacturing Message Specification (MMS): ISO 9506
		Association Control Service Element (ACSE): ISO/IEC 8650/X.227
Presentation (L6)	PPDU	OSI Connection Oriented Presentation ISO 8823/X.226
Session (L5)	SPDU	OSI Connection Oriented Session: ISO 8327/X.225
Transport (L4)	TPDU	Connection-Oriented Transport Protocol: ISO/IEC 8073/X.224
		ISO Transport over TCP (TPKT): RFC 1006
		Transmission Control Protocol (TCP): RFC 793
Network (L3)	NPDU	Internet Protocol (IP): RFC 791
Data Link (L2)	Data Frame	Ethernet: ISO/IEC 8802-3
Physical (L1)	Bits	

Figure A.6: MMS OSI model over TCP/IP

MMS protocol as defined in ISO 9506 [3] implements two types of communications:

- *Confirmed MMS Services*

  Confirmed MMS services are requested through the use of the Confirmed-Request PDU . Standard ISO 9506 defines 87 different services, e.g., getNameList, read, write, getVariableAccessAttributes.

  Each Confirmed-Request PDU is confirmed by a Confirmed-Response PDU or a Confirmed-Error PDU. In addition, the Confirmed-Request PDU can be suspended by the Cancel-Request PDU which is confirmed using a Cancel-Response PDU or a Cancel-Error PDU. Each instance of the Request PDU is correlated to the corresponding Response PDU using the InvokeID which is a 32-bit unsigned integer.

- *Unconfirmed MMS Services*

  For unconfirmed MMS services, no response PDU or error PDU will be received. Further, it is not possible to cancel an unconfirmed MMS service.

> The standard defines three different unconfirmed services: informationReport, unsolicitedStatus, and eventNotification.

The standard defines 14 types of MMS PDUs which can be easily identified by the TLV identifier with the tag number given by the standard. An example of MMS PDU is in Figure A.7.

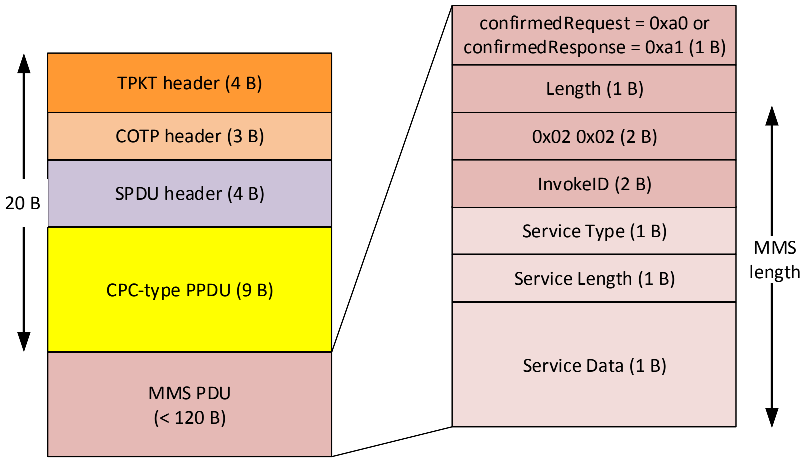


Figure A.7: MMS Confirmed-Request PDU

MMS communication can be divided into three phases: connection establishment, data initialization, and data access, see Figure A.8.

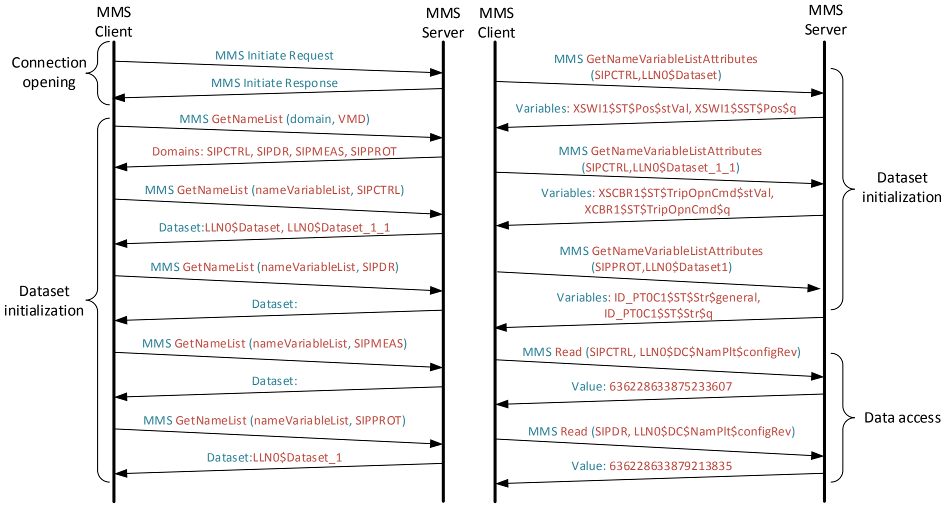

MMS
Client
MMS
Server
Connection
opening
MMS Initiate Request
MMS Initiate Response
MMS GetNameList (domain, VMD)
Domains: SIPCTRL, SIPDR, SIPMEAS, SIPPROT
MMS GetNameList (nameVariableList, SIPCTRL)
Dataset:LLN0$Dataset, LLN0$Dataset_1_1
MMS GetNameList (nameVariableList, SIPDR)
Dataset:
MMS GetNameList (nameVariableList, SIPMEAS)
Dataset:
MMS GetNameList (nameVariableList, SIPPROT)
Dataset:LLN0$Dataset_1
Dataset
initialization
MMS
Client
MMS
Server
MMS GetNameVariableListAttributes
(SIPCTRL,LLN0$Dataset)
Variables: XSWI1$ST$Pos$stVal, XSWI1$SST$Pos$q
MMS GetNameVariableListAttributes
(SIPCTRL,LLN0$Dataset_1_1)
Variables: XSCBR1$ST$TripOpnCmd$stVal,
XCBR1$ST$TripOpnCmd$q
MMS GetNameVariableListAttributes
(SIPPROT,LLN0$Dataset1)
Variables: ID_PT0C1$ST$Str$general,
ID_PT0C1$ST$Str$q
Dataset
initialization
MMS Read (SIPCTRL, LLN0$DC$NamPlt$configRev)
Value: 636228633875233607
MMS Read (SIPDR, LLN0$DC$NamPlt$configRev)
Value: 636228633879213835
Data access

Figure A.8: Example of MMS communnication

1. *Opening the communication*

   Connection is established using L4, L5, L6 and L7 protocols. On L4, the TCP establishes connection through port 102. Later, the session layer (L5) sends Connect Session Protocol Data Unit (SPDU) with Calling and Called Session Selectors. Then, the presentation layer (L6) transmits a CP Presentation Protocol Data Unit (PPDU) with a list of defined presentation contexts, e.g., ACSE abstract syntax (OID 2.2.1.0.1) with identifier 1 and MMS abstract syntax (OID 1.0.9506.1) with identifier 2. The contexts are usually encoded using BER encoding (OID 2.1.1).

   On L 7, Application Association Request (AARQ) and Application Association Response (AARE) are exchanged within MMS initiate-Request and initiate-Response messages, respectively. In these messages, connection parameters are negotiated, e.g., a number of maximal served peers, level of data structure nesting, protocol version, supported conformance parameters, and a list of supported services. The MMS initiate-Request is confirmed by the initiate-Response PDU.

2. *Data initialization*

   Following phases of communication are provided by a sequence of confirmed-Request and confirmed-Response PDUs. The requests are bound with responses using InvokeID sequence number.

   Data initialization discovers available Virtual Manufacturing Devices (VMDs) on the destination device. For each VMD, a list of logical nodes, data objects, and attributes is obtained using getNameList and getNamedVariableListAttributes services.

   During data initialization phase a client requests names of available logical nodes, datasets, variables, and attributes using getNameList, getVariableListAttributes, getNamedVariableListAttributes, etc.

3. *Data access*

   After initialization, discovered data objects are accessed for reading, writing and other operations.

MMS Monitoring

Based on analysis of several MMS datasets, it seems to be reasonable to extract following data from MMS headers:

- MMS PDU type: initateRequest, initiateResponse, confirmedRequest, confirmedResponse, unconfirmedPDU, conclude, etc.

- For confirmed and unconfirmed services it would be useful to determine the type of the service, e.g., read, getNameList, informationReport, etc. It can be also useful to know what variable is requested. However, it can be tens of variable.belonging to a logical device.

- Similarly to read service, it is interesting to detect write requests and destination datasets or variables.
- It is not feasible to extract names of datasets or variables from MMS messages unless it is specifically required. The reason is that a message may contain tens of variable in getNameList, read, write, and other services.
- Another approach to security monitoring is to monitor stations (clients) that are allowed to provide such operations and create a baseline how these operation are provided in time so that potential attacks on the communication can be detected.

A.3 IEC 61850 GOOSE

Generic Object-Oriented Substation Event (GOOSE) protocol [8] is a Ethernet-based protocols that is used for passing power measurement as well as for tripping and interlocking circuits. It implements transfer of time-critical events such as protection of electrical equipment between IEC 61850 devices [86]. GOOSE communication is based on publish-subscribe mechanism where a publisher IED sends a message to a group of subscriber IEDs through via L2 multicast. The publisher writes the values of a defined dataset into a local buffer at the sending side. The subscriber(s) reads the values, see Figure A.9.

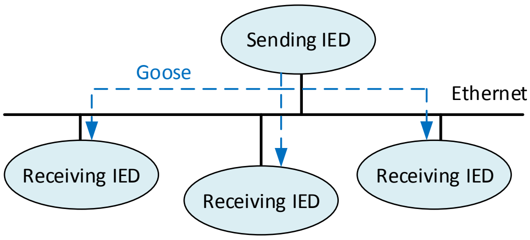


Figure A.9: GOOSE publish-subscribe communication

The GOOSE message format is shown in Figure A.10. The message consists of the source and destination MAC addresses, optional VLAN tag, followed by EtherType 0x88b8 which identifies GOOSE message. Application Identifier (APPID) identifies the receiving application. GOOSE application protocol data unit (APDU) is defined by Abstract Syntax Notation One (ASN.1) and encoded by Basic Encoding Rules (BER) [43].

On application layer, GOOSE messages is defined using ASN.1 notation, see Figure A.10. The structure of GOOSE PDU is derived from GOOSE Control Block object as defined by IEC 61850-7-2 standard [7]. It consists of the following items:

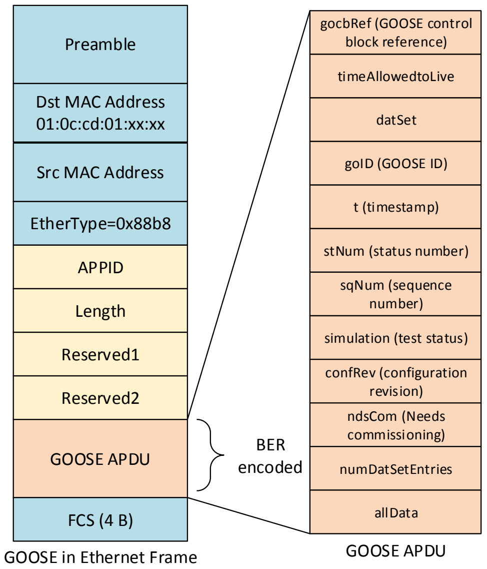


Figure A.10: GOOSE message format

- *GoCBRef:* GOOSE control block reference is a unique path-name of an instance of GOOSE Control Block (GoCB) within LLN0. The format is LDName/LLN0.GoCBName, e.g., GEDeviceF650/LLN0$GO$gcb01 where LD name is GEDeviceF650, LN class is LLN0 (Logical Node Zero), functional constraint is GO (GOOSE Control) and GoCB instance is gcb01.

- *TimeAllowedtoLive:* time at which the attribute StNum was incremented. It informs subscribers of how long to wait for the next repetition of the message.

- *DatSet:* references of the data set whose values of members shall be transmitted, e.g., GEDeviceF650/LLN0$GOOSE1. The members of the DataSet shall be uniquely numbered beginning with 1. This number is called the MemberOffset of a given member. Each member of the DataSet has a unique number and a MemberReference (the functional constraint data FCD or DataAttribute FCDA), see Figure 12.

- *GoID:* GOOSE ID is an attribute that allows a user to assign an identification for the GOOSE message, e.g., F650_GOOSE1.

- *T (timestamp):* time at which the attribute StNum was incremented.

- *StNum (status number)* is a counter that increments each time a GOOSE message has been sent and a value change has been detected within the DataSet specified by DatSet. The initital value shall be 1. The value 0 is reserved.

- *SqNum (sequence number* is the current sequence number of the reports. It shall increment each time a GOOSE message sent. Following a StNum change, the counter SqNum shall be set to a value 0. The initial value for SqNum upon a transmission of GoEna to TRUE is 1. This number seems to be similar to the sequence number in TCP.
- *Simulation (test bit):* if true, the message and therefore its value have been issued by a simulation unit and are not real values. The GOOSE subscriber will report the value of the simulated message to its application instead of the real message depending on the setting of the receiving IED.
- *ConfRev (configuration revision)* contains the configuration revision to indicate deletion of a member of the data set or the reordering of the members, or changing the DatSet reference. The number shall represent a count of the number of times that the configuration of the DataSet referenced by DatSet value has been changed.
- *NdsCom (needs commission)* indicates in the message that some commissioning is required. If TRUE, the GoCB requires further configuration.
- *NumDatSetEntries:* a number of data set entries
- *allData:* a list of user defined information of the MMS NamedVariableList that is specified in GOOSE control block.

A.4 DLMS/COSEM

DLMS (Device Language Message Specification), standard IEC 61334-4-41[1], is an application layer specification designed to support messaging to and from (energy) distribution devices. Applications like remote meter reading, remote control and value added services for metering any kind of energy, like electricity, water, gas, or heat are supported. DLMS specification is used to describe interface classes for various objects available (voltage, current) with their attributes. Detailed analysis of DLMS protocol can be found in [84].

COSEM Objects

COSEM [2] is an interface model of communicating energy metering equipment that provides a view of the functionality available through the communication interface. It provides semantics for metering application. COSEM model uses an object-oriented approach. An instance of a COSEM interface class is called *COSEM interface object*. The set of objects instantiated in the logical devices of a physical device model the functionality of the metering equipment as it is seen through its communicating interfaces.

The COSEM model represents the meter as a server used by client applications that retrieve data from, provide control information to, and instigate known actions within the meter via controlled access to the attributes and specific methods of objects making up the server interface. The client may be supporting the business processes of utilities, customers, meter operators, or meter manufacturers.

The COSEM server is structured into three hierarchical levels: physical device, logical device, and accessible COSEM objects, see Figure A.11.

- A physical device hosts one or several logical devices. A logical device models a specific functionality of the physical device. Each physical device shall contain a ”Management logical device”. For example, in a multi-energy meter, on logical device could be an electricity meter, another, a gas meter, etc.

- A logical device is a container for COSEM objects. A COSEM object is simply a structured piece of information with attributes and methods. All objects that share the same structure are of the same COSEM class. There are many COSEM classes [6, 5].

- Each logical device can be identified by its unique logical device name (LDN). This name can be retrieved from an instance of IC SAP assignment (class_id=17) or from a COSEM object COSEM logical device name of the IC Data.

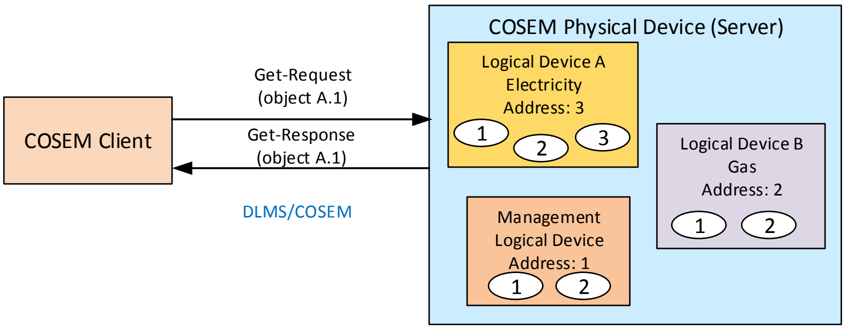


Figure A.11: Physical COSEM object with logical devices

In order to access COSEM objects in the server, an application association (AA) must be first established with a client. This identifies the partners and characterizes the context within which the association applications will communicate. The context includes:

- the application context,

- the authentication context, and

- the xDLMS context.

The information is contained in a special COSEM object Association. There are two types of the Association object depending on the name referencing: long names (LN) or short names (SN).

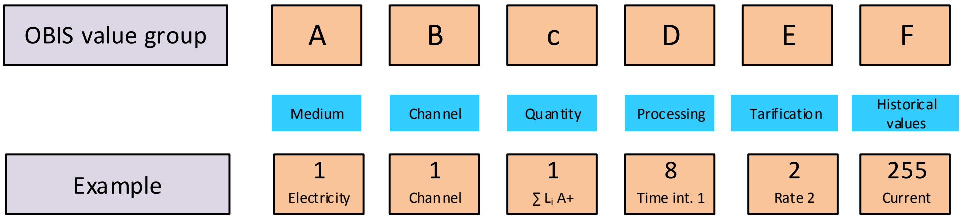
Each logical device contains at least one object of class Association LN (class_id=15) or Association SN (class_id=12). This object has an attribute no. 2 called *object_list* that contains the list of all objects available in the logical device. This helps to find which objects exist in the given logical device. The association object has predefined logical name 0.0.40.0.0.255. Therefore, all available object within a logical device can be found just by reading the object list.

Object Identification

The COSEM Object Identification System (OBIS) defines the identification codes (ID-codes) for commonly used data items in metering equipment. It provides a unique identifier for all data within the metering equipment. The ID codes defined by OBIS are used for the identification of:

- logical names of the various instances of the ICs or objects,
- data transmitted through communication lines, and
- data displayed on the metering equipment.

Example of an OBIS code is depicted at Figure A.12. The code 1.1.1.8.2. 255 represents a simple meter of electricity (A=1) on channel 1 (B=1), where the measured quantity is ΣL_iActive power+ (C=1) using time integral 1 (D=8), rate is 2 (E=2) and the current value is 255 (F).



OBIS value group	A	B	c	D	E	F
	Medium	Channel	Quantity	Processing	Tarification	Historical values
Example	1 Electricity	1 Channel	1 $\sum L_i$ A+	8 Time int. 1	2 Rate 2	255 Current

Figure A.12: OBIS data identification system

Communication

DLMS/COSEM (IEC 62056-53,62) is a standard specification using COSEM for interface modeling equipment and DLMS for data exchange of such metering equipment. It comprises the object model, the application layer protocol and the communication profiles to transport the messages.

Layer	**Function**	**DLSM/COSEM**
Application	Network process to application	Application
Presentation	Data representation, encryption and decryption, convert machine dependent data to machine independent data	COSEM
Session	Interhost communication, managing sessions between applications	DLMS
Transport	End-to-end connection, reliability and flow control	DLMS
Network	Path determination and logical addressing	DLMS
Data link	Physical addressing	HDLC, IEC 62056-47
Physical	Media, signal and binary transmission	serial media, cable, radio

Table A.1: DLMS and ISO OSI model

In ISO OSI model DLMS communicates over L4-L5 (transport and session layer), COSEM forms presentation layer (L6), see Table A.1.

Data exchange between data collection systems and metering equipment using the COSEM interface object model is based on the client/server paradigm. Metering equipment plays the role of the server. The data collection application and metering application are modelled as one or more application processes (APs). Therefore, in this environment communication takes place always between a client and a server AP: the client AP requests services and the server AP provides them. The client AP can exchange data with single or multiple server APs at the same time. The server AP can exchange data with one or more client APs at the same time.

The request can be ”read the object 1.0.1.8.0.255” and the answer is ”1789.8 kWh”. Before being able to send requests, the client has first to establish a connection with the other side using Association Control Service Element (ACSE) data exchange:

1. *Session establishment.*

   Session establishment and tear-down uses ACSE defined by ITU-T X.227 [63]. Figure A.13 shows ACSE messages (blue) and DLMS/ COSEM Application Protocol Data Units (APDUs, black).

   The ACSE protocol establishes and tears down an Application Association (AA) between a client Application Process and a server Application Process for a given Application Context Name. This name identifies whether an encrypted or plaintext application context is being requested and whether short name or long names are being used for addressing.

2. *Data Exchange.*

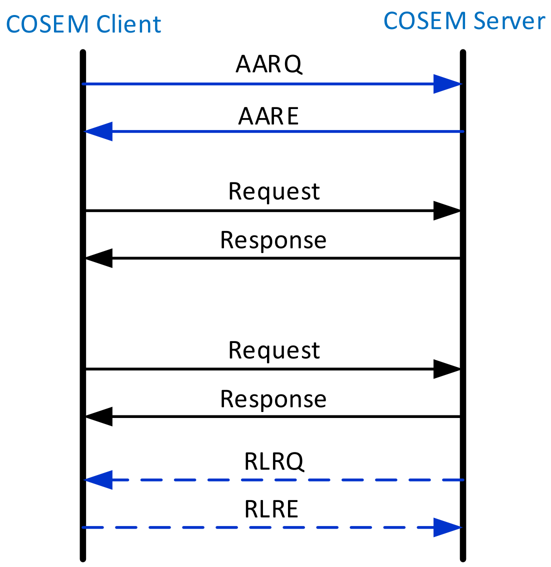


Figure A.13: Opening COSEM communication

After establishing association, protocol DLMS/COSEM can read data, write data or invoke actions. DLMS/COSEM APDU format is also variable and encoded using A-XDR rules [4]. Example of DLMS Get-Request and Get-Response APDUs is in Figure A.14.

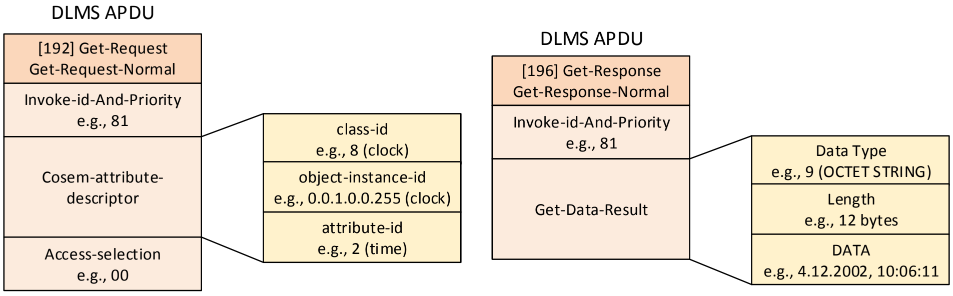


Figure A.14: DLMS/COSEM requests and responses

Appendix B

IPFIX Templates for ICS Protocols

IP Flow Information Export (IPFIX) is a flexible format for the exchange of flow information obtained during flow monitoring. The standard specifies the IPFIX protocol by RFC 7011 [34] and the information model by RFC 7012 [33]. A great advantage of IPFIX specification is that it allows to extend the IPFIX information model by adding new information elements retrieved by advanced flow monitoring probes.

In case of ICS flow monitoring, it is possible to define an ICS flow template for each of the ICS flow protocol so that specific ICS data are collected by IPFIX monitoring. Here we provide few examples of ICS templates for selected ICS protocols deployed in smart grid networks.

B.1 IPFIX Flow Template for IEC 104

Standard IPFIX information elements are defined by IANA[1]. For common information elements, the IANA defines a name of the element, its identifier, abstract data type and semantics. An example of standardized IPFIX information elements is given in Table B.1.

As mentioned in Section 3.4.2, for IEC 104 analysis we identify six IEC 104 header fields, see Figure 3.4 on page 42, that are important for higher IEC 104 visibility and anomaly detection. These fields include the APDU frame type, APDU length, ASDU type, ASDU cause of transmission, the number of information objects transmitted within the ASDU, originator address, and ASDU address.

For these selected IEC 104 headers, we need to define new IPFIX fields for the IPFIX template. IPFIX definition includes definition of the information element name, its ID, data type and the length. Table B.2 shows user-defined IEC 104 information elements as proposed by this research.

[1] See `https://www.iana.org/assignments/ipfix/ipfix.xhtml` [August 2011].

ID	Name	Data Type	Description
1	Octets	unsigned64	The number of octets since the previous report.
2	Packets	unsigned64	The number of incoming packets since the previous report.
4	Protocol	unsigned8	The value of the protocol number in the IP packet header.
5	ToS	unsigned8	The value of the TOS field in the IPv4 packet header.
6	TCP flags	unsigned16	TCP control bits observed for the packets of this Flow.
7	SrcPort	unsigned16	The source port identifier in the transport header.
8	SrcIP	ipv4Address	The IPv4 source address in the IP packet header.
9	SrcPrefix	unsigned8	The number of contiguous bits that are relevant in the sourceIPv4Prefix.
10	Ingress	unsigned32	The index of the IP interface where packets of this Flow are being received.
11	DstPort	unsigned16	The destination port identifier in the transport header.
12	DstIP	ipv4Address	The IPv4 destination address in the IP packet header.
...	...	...	...

Table B.1: Example of the standardized IPFIX information elements.

.

Figure B.1 shows an example of the IPFIX template for IEC 104 and the corresponding flow record with meta data about the IEC 104 virtual flow. The template record transmits information about the used template. It is regularly sent by an IPFIX monitoring probe to inform the IPFIX collector how to interpret received flow records. In this case the IEC 104 template with ID=257 includes standardized information elements as defined by RFC 7012 [33], e.g., the number of transmitted bytes, packets, IP protocol number, TCP flags, source and destination IP addresses, TCP ports, etc. Information elements with ID 370-376 are ICS-specific fields that transmit data about IEC 104 flows.

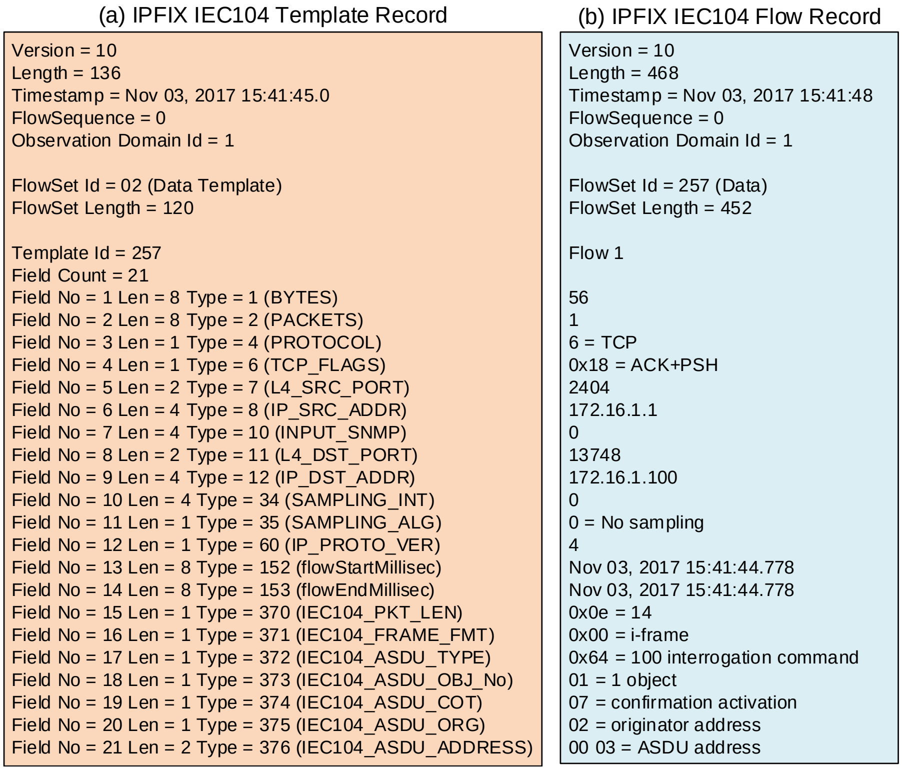
The figure also show an IEC 104 flow record that transmits monitoring data about observed IEC 104 flow. The IPFIX flow record was created using IPFIX template with ID=257, see the item FlowSetId and contains meta information about the IEC 104 flow that was captured on Nov 03, 2017 at 15:41:47 at the monitoring probe. The flow was sent from station 172.16.1.1, port 2404 to station 172.16.100, port 13748 and contained only one packet of size 56 bytes. The packet included i-frame ASDU of 14 bytes with ASDU type=100 (interrogation command) and ASDU CoT=07 (confirmation activation). The flow record also includes originator address (2) and ASDU address (3).

ID	Name	Data Type	Length (B)
370	IEC104_PKT_LENGTH	unsigned8	1
371	IEC140_FRAME_FMT	unsigned8	1
372	IEC140_ASDU_TYPE	unsigned8	1
373	IEC140_ASDU_OBJ_COUNT	unsigned8	1
374	IEC140_ASDU_COT	unsigned8	1
375	IEC140_ASDU_ORG	unsigned8	1
376	IEC140_ASDU_ADDRESS	unsigned16	2

Table B.2: New information elements for IEC 104.

A big advantage of IPFIX flow monitoring is the fact that IPFIX flow records include only values of the observed flow, not fields. Thus, the flow records represent a compact data structure for transmitting important monitoring data.

Figures B.2 and B.3 show an example of IEC 104 packet and an IEC 104 flow record in Wireshark. We can notice that IEC 104 protocol is encapsulated in TCP and one TCP packet may include multiple IEC 104 ASDU messages. This is the reason why we split TCP messages into single ASDUs represented by virtual flows. This increases visibility of IEC 104 communication as discussed in Sec. 3.6, p. 49.



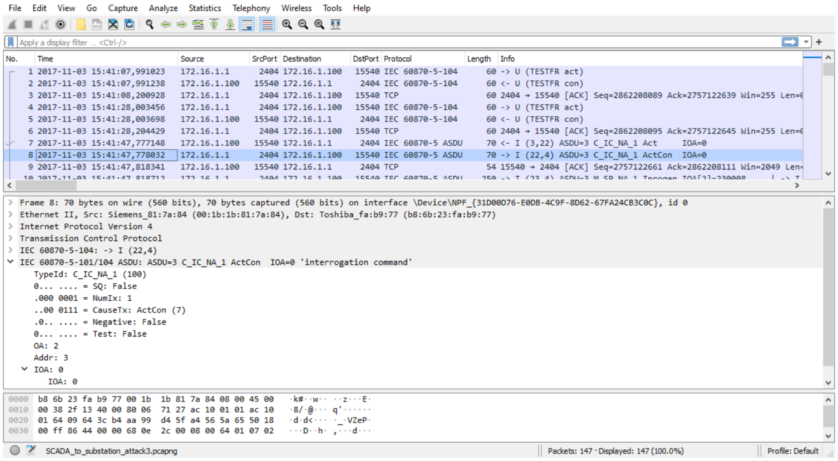
(a) IPFIX IEC104 Template Record

```
Version = 10
Length = 136
Timestamp = Nov 03, 2017 15:41:45.0
FlowSequence = 0
Observation Domain Id = 1

FlowSet Id = 02 (Data Template)
FlowSet Length = 120

Template Id = 257
Field Count = 21
Field No = 1 Len = 8 Type = 1 (BYTES)
Field No = 2 Len = 8 Type = 2 (PACKETS)
Field No = 3 Len = 1 Type = 4 (PROTOCOL)
Field No = 4 Len = 1 Type = 6 (TCP_FLAGS)
Field No = 5 Len = 2 Type = 7 (L4_SRC_PORT)
Field No = 6 Len = 4 Type = 8 (IP_SRC_ADDR)
Field No = 7 Len = 4 Type = 10 (INPUT_SNMP)
Field No = 8 Len = 2 Type = 11 (L4_DST_PORT)
Field No = 9 Len = 4 Type = 12 (IP_DST_ADDR)
Field No = 10 Len = 4 Type = 34 (SAMPLING_INT)
Field No = 11 Len = 1 Type = 35 (SAMPLING_ALG)
Field No = 12 Len = 1 Type = 60 (IP_PROTO_VER)
Field No = 13 Len = 8 Type = 152 (flowStartMillisec)
Field No = 14 Len = 8 Type = 153 (flowEndMillisec)
Field No = 15 Len = 1 Type = 370 (IEC104_PKT_LEN)
Field No = 16 Len = 1 Type = 371 (IEC104_FRAME_FMT)
Field No = 17 Len = 1 Type = 372 (IEC104_ASDU_TYPE)
Field No = 18 Len = 1 Type = 373 (IEC104_ASDU_OBJ_No)
Field No = 19 Len = 1 Type = 374 (IEC104_ASDU_COT)
Field No = 20 Len = 1 Type = 375 (IEC104_ASDU_ORG)
Field No = 21 Len = 2 Type = 376 (IEC104_ASDU_ADDRESS)
```

(b) IPFIX IEC104 Flow Record

```
Version = 10
Length = 468
Timestamp = Nov 03, 2017 15:41:48
FlowSequence = 0
Observation Domain Id = 1

FlowSet Id = 257 (Data)
FlowSet Length = 452

Flow 1

56
1
6 = TCP
0x18 = ACK+PSH
2404
172.16.1.1
0
13748
172.16.1.100
0
0 = No sampling
4
Nov 03, 2017 15:41:44.778
Nov 03, 2017 15:41:44.778
0x0e = 14
0x00 = i-frame
0x64 = 100 interrogation command
01 = 1 object
07 = confirmation activation
02 = originator address
00 03 = ASDU address
```

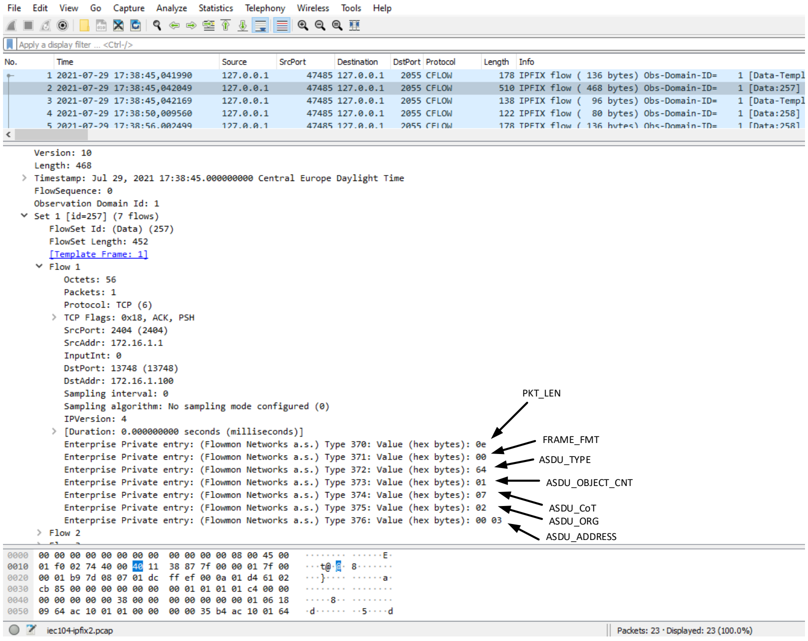
Figure B.1: An IPFIX template record (a) and flow record for IEC104 (b)


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time                        Source        SrcPort Destination  DstPort Protocol          Length Info
   1 2017-11-03 15:41:07,991023  172.16.1.1       2404 172.16.1.100   15540 IEC 60870-5-104       60 -> U (TESTFR act)
   2 2017-11-03 15:41:07,991238  172.16.1.100    15540 172.16.1.1      2404 IEC 60870-5-104       60 <- U (TESTFR con)
   3 2017-11-03 15:41:08,200928  172.16.1.1       2404 172.16.1.100   15540 TCP                   60 2404 → 15540 [ACK] Seq=2862208089 Ack=2757122639 Win=255 Len=
   4 2017-11-03 15:41:28,003456  172.16.1.1       2404 172.16.1.100   15540 IEC 60870-5-104       60 -> U (TESTFR act)
   5 2017-11-03 15:41:28,003698  172.16.1.100    15540 172.16.1.1      2404 IEC 60870-5-104       60 <- U (TESTFR con)
   6 2017-11-03 15:41:28,204429  172.16.1.1       2404 172.16.1.100   15540 TCP                   60 2404 → 15540 [ACK] Seq=2862208095 Ack=2757122645 Win=255 Len=
   7 2017-11-03 15:41:47,777148  172.16.1.100    15540 172.16.1.1      2404 IEC 60870-5 ASDU      70 <- I (3,22) ASDU=3 C_IC_NA_1 Act     IOA=0
   8 2017-11-03 15:41:47,778032  172.16.1.1       2404 172.16.1.100   15540 IEC 60870-5 ASDU      70 -> I (22,4) ASDU=3 C_IC_NA_1 ActCon  IOA=0
   9 2017-11-03 15:41:47,818341  172.16.1.100    15540 172.16.1.1      2404 TCP                   54 15540 → 2404 [ACK] Seq=2757122661 Ack=2862208111 Win=2049 Len=
  10 2017-11-03 15:41:47,818712  172.16.1.1       2404 172.16.1.100   15540 IEC 60870-5 ASDU     250 -> I (23,4) ASDU=3 M_SP_NA_1 Inrogen IOA[2]=230008     | -> I

> Frame 8: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{31D00D76-E0DB-4C9F-8D62-67FA24CB3C0C}, id 0
> Ethernet II, Src: Siemens_81:7a:84 (00:1b:1b:81:7a:84), Dst: Toshiba_fa:b9:77 (b8:6b:23:fa:b9:77)
> Internet Protocol Version 4
> Transmission Control Protocol
> IEC 60870-5-104: -> I (22,4)
v IEC 60870-5-101/104 ASDU: ASDU=3 C_IC_NA_1 ActCon  IOA=0 'interrogation command'
    TypeId: C_IC_NA_1 (100)
    0... .... = SQ: False
    .000 0001 = NumIx: 1
    ..00 0111 = CauseTx: ActCon (7)
    .0.. .... = Negative: False
    0... .... = Test: False
    OA: 2
    Addr: 3
  v IOA: 0
      IOA: 0

0000  b8 6b 23 fa b9 77 00 1b  1b 81 7a 84 08 00 45 00   ·k#··w·· ··z···E·
0010  00 38 2f 13 40 00 80 06  71 27 ac 10 01 01 ac 10   ·8/·@··· q'······
0020  01 64 09 64 3c b4 aa 99  d4 5f a4 56 5a 65 50 18   ·d·d<··· ·_·VZeP·
0030  00 ff 86 44 00 00 68 0e  2c 00 08 00 64 01 07 02   ···D··h· ,···d···

SCADA_to_substation_attack3.pcapng        Packets: 147 · Displayed: 147 (100.0%)        Profile: Default
```

Figure B.2: Analysis of an IEC 104 packet in Wireshark

Figure B.3 depicts an example of an IPFIX flow record with specific IEC 104 related fields. From that flow we can see, that the station with source IP address 172.16.1.1 and port 2404 (IEC 104 server) sent a message to 172.16.1.100 (IEC 104 client) an interrogation command (ASDU type 100, i.e., 0x64) and the cause of transmission 07 (confirmation activation). As mentioned in the previous text, such detailed monitoring information enable high visibility of IEC 104 traffic in the smart grid which is important for security monitoring and anomaly detection.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
PKT_LEN
FRAME_FMT
ASDU_TYPE
ASDU_OBJECT_CNT
ASDU_CoT
ASDU_ORG
ASDU_ADDRESS

```
No.  Time                        Source     SrcPort Destination DstPort Protocol  Length Info
   1 2021-07-29 17:38:45,041990  127.0.0.1    47485 127.0.0.1      2055 CFLOW        178 IPFIX flow ( 136 bytes) Obs-Domain-ID=    1 [Data-Templ
   2 2021-07-29 17:38:45,042049  127.0.0.1    47485 127.0.0.1      2055 CFLOW        510 IPFIX flow ( 468 bytes) Obs-Domain-ID=    1 [Data:257]
   3 2021-07-29 17:38:45,042169  127.0.0.1    47485 127.0.0.1      2055 CFLOW        138 IPFIX flow (  96 bytes) Obs-Domain-ID=    1 [Data-Templ
   4 2021-07-29 17:38:50,009560  127.0.0.1    47485 127.0.0.1      2055 CFLOW        122 IPFIX flow (  80 bytes) Obs-Domain-ID=    1 [Data:258]
   5 2021-07-29 17:38:56,002499  127.0.0.1    47485 127.0.0.1      2055 CFLOW        178 IPFIX flow ( 136 bytes) Obs-Domain-ID=    1 [Data:258]

    Version: 10
    Length: 468
  > Timestamp: Jul 29, 2021 17:38:45.000000000 Central Europe Daylight Time
    FlowSequence: 0
    Observation Domain Id: 1
  v Set 1 [id=257] (7 flows)
      FlowSet Id: (Data) (257)
      FlowSet Length: 452
      [Template Frame: 1]
    v Flow 1
        Octets: 56
        Packets: 1
        Protocol: TCP (6)
      > TCP Flags: 0x18, ACK, PSH
        SrcPort: 2404 (2404)
        SrcAddr: 172.16.1.1
        InputInt: 0
        DstPort: 13748 (13748)
        DstAddr: 172.16.1.100
        Sampling interval: 0
        Sampling algorithm: No sampling mode configured (0)
        IPVersion: 4
      > [Duration: 0.000000000 seconds (milliseconds)]
        Enterprise Private entry: (Flowmon Networks a.s.) Type 370: Value (hex bytes): 0e
        Enterprise Private entry: (Flowmon Networks a.s.) Type 371: Value (hex bytes): 00
        Enterprise Private entry: (Flowmon Networks a.s.) Type 372: Value (hex bytes): 64
        Enterprise Private entry: (Flowmon Networks a.s.) Type 373: Value (hex bytes): 01
        Enterprise Private entry: (Flowmon Networks a.s.) Type 374: Value (hex bytes): 07
        Enterprise Private entry: (Flowmon Networks a.s.) Type 375: Value (hex bytes): 02
        Enterprise Private entry: (Flowmon Networks a.s.) Type 376: Value (hex bytes): 00 03
    > Flow 2

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  01 f0 02 74 40 00 40 11  38 87 7f 00 00 01 7f 00   ···t@·@· 8·······
0020  00 01 b9 7d 08 07 01 dc  ff ef 00 0a 01 d4 61 02   ···}···· ······a·
0030  cb 85 00 00 00 00 00 00  00 01 01 01 01 c4 00 00   ········ ········
0040  00 00 00 00 00 38 00 00  00 00 00 00 00 01 06 18   ·····8·· ········
0050  09 64 ac 10 01 01 00 00  00 00 35 b4 ac 10 01 64   ·d······ ··5····d

iec104-ipfix2.pcap        Packets: 23 · Displayed: 23 (100.0%)
```

Figure B.3: IPFIX record with the IEC 104 fields in Wireshark

B.2 IPFIX Flow Template for MMS

IEC 61850 Manufacturing Message Specification (MMS) protocol describes high-level communication between real industrial devices. Standards ISO/ IEC 9506-1 and 2 [3] specify the MMS services and the protocol. The MMS system communicates using the client-server model where the client is a network application or device (e.g., control center) that asks for data or an action from the server. The server contains so called *Virtual Manufacturing Device* (VMD) that models processes and actions implemented on the server device using a collection of MMS objects. Practically, the client requests a specific operation on the given MMS object via MMS protocol.

For security monitoring, it is important to observe exchanged MMS messages and see what operations on what objects are requested on a given station. As described in Appendix A.2, the MMS protocols is an application layer industrial protocol. The protocol was originally defined for OSI stack, thus its implementation over TCP/IP includes multiple sub-layers encapsulated within L4 and L7 layer, see Figure A.6, page 130.

To obtain a required granularity of MMS visibility as mentioned in Section 3.6 on page 49, we propose to create a virtual MMS flow for each transmitted MMS packet, similarly as in case of the IEC 104. This obtains packet-level visibility of MMS communication that is important for anomaly detection.

The MMS IPFIX template gathers the following data from MMS packets:

- *MMS type* defines a type of MMS PDU which includes 14 types [86], e.g., confirmed-requestPDU (type=0), confirmed-responsePDU (type= 1), confirmed-errorPDU (type=2), unconfirmed-PDU (type=3), rejectPDU (type=4), etc.

- *MMS confirmed service request* is a specific description of MMS type=1 that contains an operation requested using the confirmed service. Confirmed services include status request (0), getNameList request (1), identify request (2), rename request (3), read request (4), write request (5), getVariableAccessAttributes request (6), etc.

- *MMS confirmed service response* contains a response value of the requested confirmed operation. The MMS confirmed service response should match previously sent MMS confirmed service request. The response packet contains the response type which has the same values as the MMS confirmed request messages and requested data. For MMS visibility, we do not monitor requested data but only the value of the service.

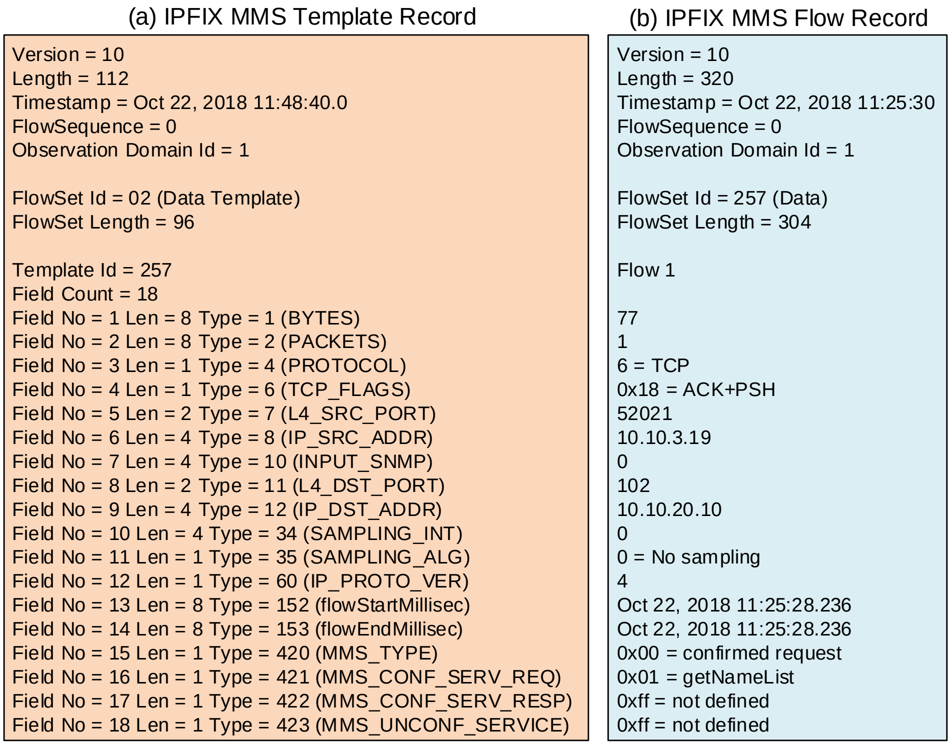
- *MMS unconfirmed service type* transmits specific details for MMS type=3 (unconfirmed-PDU), more specifically the type of unconfirmed message which is informationReport (0), unsolicitedStatus (1), or eventNotification (2).

ID	Name	Data Type	Length (B)
420	MMS_TYPE	unsigned8	1
421	MMS_CONF_SERVICE_REQ	unsigned8	1
422	MMS_CONF_SERVICE_RESP	unsigned8	1
423	MMS_UNCONF_SERVICE	unsigned8	1

Table B.3: New information elements for MMS.

The application layer fields defined for the IPFIX MMS template are given in Table B.3. Comparing to the IEC 104 or GOOSE protocols, the MMS template is more compact, nevertheless, it includes important meta data about MMS communication. An example of the MMS template record and MMS flow record is in Figure B.4.

The flow record in Figure B.4 (b) transmit information about MMS message sent from the client on IP address 10.10.3.19 to the server (VMD) at address 10.10.20.10. The client requested a list of all available MMS objects on the device. This is a standard operation when a connection between the MMS client and server is established. On the other hand, this message may also indicate the recon-



(a) IPFIX MMS Template Record

```
Version = 10
Length = 112
Timestamp = Oct 22, 2018 11:48:40.0
FlowSequence = 0
Observation Domain Id = 1

FlowSet Id = 02 (Data Template)
FlowSet Length = 96

Template Id = 257
Field Count = 18
Field No = 1 Len = 8 Type = 1 (BYTES)
Field No = 2 Len = 8 Type = 2 (PACKETS)
Field No = 3 Len = 1 Type = 4 (PROTOCOL)
Field No = 4 Len = 1 Type = 6 (TCP_FLAGS)
Field No = 5 Len = 2 Type = 7 (L4_SRC_PORT)
Field No = 6 Len = 4 Type = 8 (IP_SRC_ADDR)
Field No = 7 Len = 4 Type = 10 (INPUT_SNMP)
Field No = 8 Len = 2 Type = 11 (L4_DST_PORT)
Field No = 9 Len = 4 Type = 12 (IP_DST_ADDR)
Field No = 10 Len = 4 Type = 34 (SAMPLING_INT)
Field No = 11 Len = 1 Type = 35 (SAMPLING_ALG)
Field No = 12 Len = 1 Type = 60 (IP_PROTO_VER)
Field No = 13 Len = 8 Type = 152 (flowStartMillisec)
Field No = 14 Len = 8 Type = 153 (flowEndMillisec)
Field No = 15 Len = 1 Type = 420 (MMS_TYPE)
Field No = 16 Len = 1 Type = 421 (MMS_CONF_SERV_REQ)
Field No = 17 Len = 1 Type = 422 (MMS_CONF_SERV_RESP)
Field No = 18 Len = 1 Type = 423 (MMS_UNCONF_SERVICE)
```

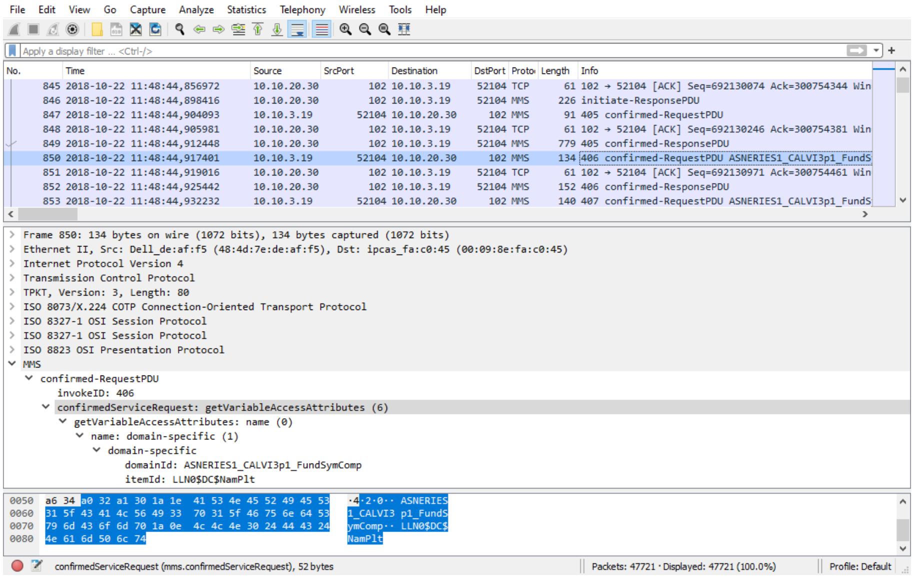
(b) IPFIX MMS Flow Record

```
Version = 10
Length = 320
Timestamp = Oct 22, 2018 11:25:30
FlowSequence = 0
Observation Domain Id = 1

FlowSet Id = 257 (Data)
FlowSet Length = 304

Flow 1

77
1
6 = TCP
0x18 = ACK+PSH
52021
10.10.3.19
0
102
10.10.20.10
0
0 = No sampling
4
Oct 22, 2018 11:25:28.236
Oct 22, 2018 11:25:28.236
0x00 = confirmed request
0x01 = getNameList
0xff = not defined
0xff = not defined
```

Figure B.4: An IPFIX template record (a) and flow record for MMS (b)

naissance attack when an attacker tries to enumerate all available objects on active MMS devices. Thus, it is important to observe communication context which can be implemented using probabilistic automata as presented in Chapter 4.

The following Figure B.5 and B.6 presents an example of MMS packet analysis in Wireshark and an example of MMS IPFIX flow record.



```
No.   Time                        Source        SrcPort Destination  DstPort Proto Length Info
  845 2018-10-22 11:48:44,856972  10.10.20.30       102 10.10.3.19     52104 TCP       61 102 → 52104 [ACK] Seq=692130074 Ack=300754344 Win
  846 2018-10-22 11:48:44,898416  10.10.20.30       102 10.10.3.19     52104 MMS      226 initiate-ResponsePDU
  847 2018-10-22 11:48:44,904093  10.10.3.19      52104 10.10.20.30      102 MMS       91 405 confirmed-RequestPDU
  848 2018-10-22 11:48:44,905981  10.10.20.30       102 10.10.3.19     52104 TCP       61 102 → 52104 [ACK] Seq=692130246 Ack=300754381 Win
  849 2018-10-22 11:48:44,912448  10.10.20.30       102 10.10.3.19     52104 MMS      779 405 confirmed-ResponsePDU
  850 2018-10-22 11:48:44,917401  10.10.3.19      52104 10.10.20.30      102 MMS      134 406 confirmed-RequestPDU ASNERIES1_CALVI3p1_FundS
  851 2018-10-22 11:48:44,919016  10.10.20.30       102 10.10.3.19     52104 TCP       61 102 → 52104 [ACK] Seq=692130971 Ack=300754461 Win
  852 2018-10-22 11:48:44,925442  10.10.20.30       102 10.10.3.19     52104 MMS      152 406 confirmed-ResponsePDU
  853 2018-10-22 11:48:44,932232  10.10.3.19      52104 10.10.20.30      102 MMS      140 407 confirmed-RequestPDU ASNERIES1_CALVI3p1_FundS
```

```
> Frame 850: 134 bytes on wire (1072 bits), 134 bytes captured (1072 bits)
> Ethernet II, Src: Dell_de:af:f5 (48:4d:7e:de:af:f5), Dst: ipcas_fa:c0:45 (00:09:8e:fa:c0:45)
> Internet Protocol Version 4
> Transmission Control Protocol
> TPKT, Version: 3, Length: 80
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8823 OSI Presentation Protocol
v MMS
  v confirmed-RequestPDU
      invokeID: 406
    v confirmedServiceRequest: getVariableAccessAttributes (6)
      v getVariableAccessAttributes: name (0)
        v name: domain-specific (1)
          v domain-specific
              domainId: ASNERIES1_CALVI3p1_FundSymComp
              itemId: LLN0$DC$NamPlt
```

```
0050  a6 34 a0 32 a1 30 1a 1e  41 53 4e 45 52 49 45 53   ·4·2·0·· ASNERIES
0060  31 5f 43 41 4c 56 49 33  70 31 5f 46 75 6e 64 53   1_CALVI3 p1_FundS
0070  79 6d 43 6f 6d 70 1a 0e  4c 4c 4e 30 24 44 43 24   ymComp·· LLN0$DC$
0080  4e 61 6d 50 6c 74                                  NamPlt
```

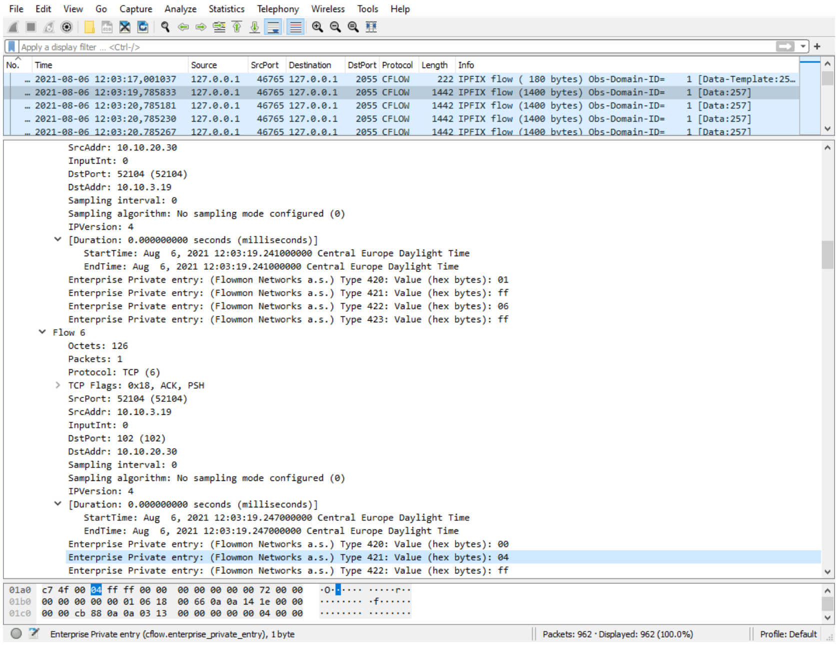
confirmedServiceRequest (mms.confirmedServiceRequest), 52 bytes || Packets: 47721 · Displayed: 47721 (100.0%) || Profile: Default

Figure B.5: Analysis of a MMS packet in Wireshark

Figure B.5 also shows how MMS messages are encapsulated using OSI stack over TCP/IP. The MMS PDU is encapsulated in the presentation protocol ISO 8823/X.226 [64], session protocols ISO 8327/X.225 [65], Connection-oriented Transport Protocol (COTP) [66] and ISO Transport Protocol over TCP (TPKT) [104]. Further information about parsing MMS messages are given in [86].

Figure B.6 depicts an example of MMS flow monitoring records. We can see two exported records as flows number 5 and 6. The first flow record includes one MMS packet sent by the MMS client with IP 10.10.20.30 to the MMS server with IP address 10.10.3.19. MMS related flow fields reveal the confirmed response PDU with operation *getVariableAccessAttributes* (6). After getting all variable attributes, the client request the one of the attributes by *read* operations (MMS type=00, MMS confirmed service request = 04).

Such MMS flow data records provide high-visibility of smart grid communication using MMS protocol and represent input data for modeling MMS communication using automata, as presented in Chapter 4.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.  Time                        Source     SrcPort Destination DstPort Protocol Length Info
 …   2021-08-06 12:03:17,001037  127.0.0.1  46765   127.0.0.1   2055    CFLOW     222   IPFIX flow ( 180 bytes) Obs-Domain-ID=    1 [Data-Template:25…
 …   2021-08-06 12:03:19,785833  127.0.0.1  46765   127.0.0.1   2055    CFLOW    1442   IPFIX flow (1400 bytes) Obs-Domain-ID=    1 [Data:257]
 …   2021-08-06 12:03:20,785181  127.0.0.1  46765   127.0.0.1   2055    CFLOW    1442   IPFIX flow (1400 bytes) Obs-Domain-ID=    1 [Data:257]
 …   2021-08-06 12:03:20,785230  127.0.0.1  46765   127.0.0.1   2055    CFLOW    1442   IPFIX flow (1400 bytes) Obs-Domain-ID=    1 [Data:257]
 …   2021-08-06 12:03:20,785267  127.0.0.1  46765   127.0.0.1   2055    CFLOW    1442   IPFIX flow (1400 bytes) Obs-Domain-ID=    1 [Data:257]
```

```
        SrcAddr: 10.10.20.30
        InputInt: 0
        DstPort: 52104 (52104)
        DstAddr: 10.10.3.19
        Sampling interval: 0
        Sampling algorithm: No sampling mode configured (0)
        IPVersion: 4
      v [Duration: 0.000000000 seconds (milliseconds)]
          StartTime: Aug  6, 2021 12:03:19.241000000 Central Europe Daylight Time
          EndTime: Aug  6, 2021 12:03:19.241000000 Central Europe Daylight Time
        Enterprise Private entry: (Flowmon Networks a.s.) Type 420: Value (hex bytes): 01
        Enterprise Private entry: (Flowmon Networks a.s.) Type 421: Value (hex bytes): ff
        Enterprise Private entry: (Flowmon Networks a.s.) Type 422: Value (hex bytes): 06
        Enterprise Private entry: (Flowmon Networks a.s.) Type 423: Value (hex bytes): ff
    v Flow 6
        Octets: 126
        Packets: 1
        Protocol: TCP (6)
      > TCP Flags: 0x18, ACK, PSH
        SrcPort: 52104 (52104)
        SrcAddr: 10.10.3.19
        InputInt: 0
        DstPort: 102 (102)
        DstAddr: 10.10.20.30
        Sampling interval: 0
        Sampling algorithm: No sampling mode configured (0)
        IPVersion: 4
      v [Duration: 0.000000000 seconds (milliseconds)]
          StartTime: Aug  6, 2021 12:03:19.247000000 Central Europe Daylight Time
          EndTime: Aug  6, 2021 12:03:19.247000000 Central Europe Daylight Time
        Enterprise Private entry: (Flowmon Networks a.s.) Type 420: Value (hex bytes): 00
        Enterprise Private entry: (Flowmon Networks a.s.) Type 421: Value (hex bytes): 04
        Enterprise Private entry: (Flowmon Networks a.s.) Type 422: Value (hex bytes): ff
```

```
01a0  c7 4f 00 04 ff ff 00 00  00 00 00 00 00 72 00 00   ·O······ ·····r··
01b0  00 00 00 00 00 01 06 18  00 66 0a 0a 14 1e 00 00   ········ ·f······
01c0  00 00 cb 88 0a 0a 03 13  00 00 00 00 00 04 00 00   ········ ········
```

```
Enterprise Private entry (cflow.enterprise_private_entry), 1 byte    Packets: 962 · Displayed: 962 (100.0%)    Profile: Default
```

Figure B.6: IPFIX record with the MMS fields in Wireshark

B.3 IPFIX Flow Template for GOOSE

IEC 61850 Generic Object-Oriented Substation Event (GOOSE) protocol [8] is designed as a publisher/subscriber type communication, see Appendix A.3. The publisher periodically sends messages and when an event occurs within any GOOSE dataset element (e.g., tripping signal, circuit breaker closing), it will stop the existing GOOSE re-transmission message. A state number contained in the GOOSE protocol identifies whether a GOOSE message is new message or has been retransmitted.

Because the protocol is publisher/subscriber based, there is no confirmation that the sent message is correctly received by the subscriber, so the message burst minimizes the chance of message loss. GOOSE data is directly embedded into Ethernet data packets and sent by the publisher on multicast or broadcast MAC addresses.

All messages are published under a topic. The subscriber receives all messages from the system, but filters and parses only the messages with the topic it has subscribed to.

Based on our analysis of GOOSE behavior [86], we identified the following fields from the GOOSE packet header that seems to be useful for security monitoring:

- *Application ID* (APPID) refers to the sending application. In include the type (2 bits) and an application identifier (14 bits). For typical GOOSE communication, the type value is *00*.

- *GOOSE Control Block Reference* (GoCBRef) is a unique path-name of an instance of GOOSE Control Block within the logical node. The control block manages processes at the publisher's side. It contains the parameter that specifies how to send the data set.
- *Data Set* name (DatSet) refer to a data set whose values of members shall be transmitted. The members of the DatSet shall be uniquely numbered beginning with 1. A GOOSE data set is a collection of data attributes transmitted in the payload of the GOOSE packet. Data set is not structured transmits uninterpreted values that are relevant for the given data set.
- *GOOSE ID* (GoID) is an attribute that allows a user to assign an identification for the GOOSE message.
- *Status Number* (StNum) is a counter that increments each time a GOOSE message has been sent and a value change has been detected within the DataSet.

For monitoring purposes it is not necessary to collect monitoring data about individual GOOSE packets because the GOOSE protocol operates as a keep-alive mechanism and consequent messages within the same subscriber group are transmitted repeatedly every few seconds without any significant changes in the packet header. Until an event happens at the publisher's side, only the sequence number (sqNum) is incremented in the GOOSE packets.

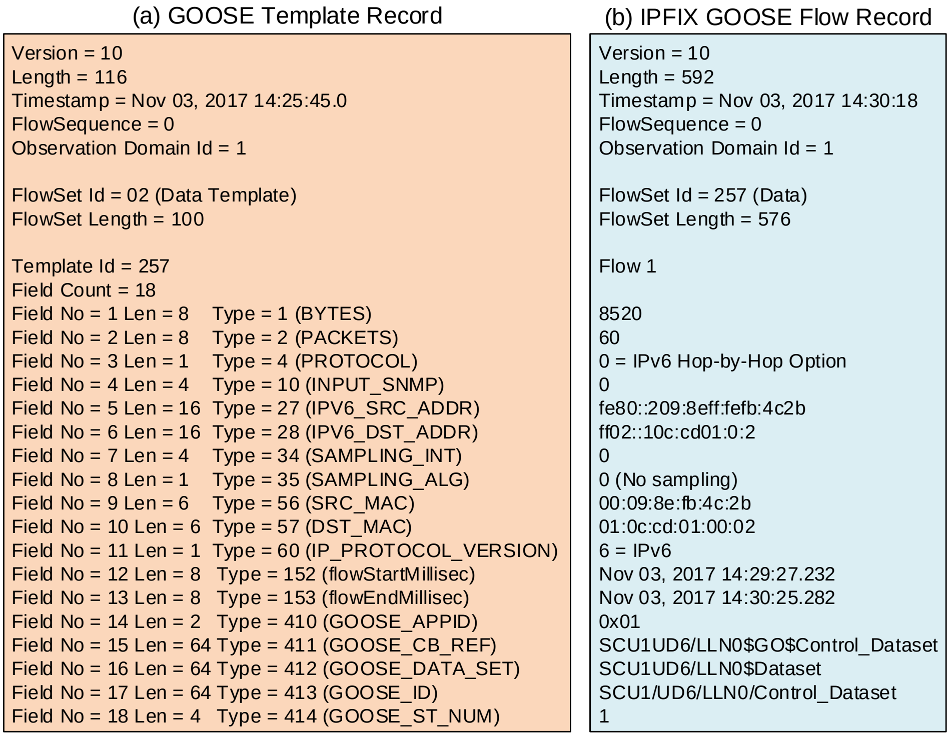
Since GOOSE operates directly on Layer 2 and IPFIX flow monitoring requires IP addresses as key items in the flow record, we create so called virtual flows that encapsulates L2 flows withing IPv6 virtual connection, see 3.4.3 on page 44. Here we transform a source unicast MAC address to a link-local IPv6 address using EUI-64 transformation [54, Appendix A] and a destination multicast MAC address to a local multicast IPv6 address by a simple transformation, where an original MAC address is prefixed by the *ff02::* IPv6 multicast prefix for link-local scope.

For the above mentioned GOOSE headers, we define the IPFIX GOOSE template, see Table B.4.

ID	Name	Data Type	Length (B)
410	GOOSE_APPID	unsigned16	2
411	GOOSE_CB_REF	char[64]	64
412	GOOSE_DATA_SET	char[64]	64
413	GOOSE_ID	char[64]	64
414	GOOSE_ST_NUM	unsigned32	4

Table B.4: New information elements for GOOSE.

Unlike IEC 104 protocol, where each IEC 104 ASDU creates a new IEC 104 flow record, definition of the GOOSE flow employs packet aggregation, i.e., GOOSE packets with the same source and destination MAC address, GOOSE Application ID, Control Block Reference, Data Set name, GOOSE ID and Status numbers define a GOOSE flow. Thus, GOOSE packet where no fields except the sequence number is changed are aggregated into one flow. The GOOSE template record and an example of GOOSE flow record are depicted in Figure B.7.



(a) GOOSE Template Record

```
Version = 10
Length = 116
Timestamp = Nov 03, 2017 14:25:45.0
FlowSequence = 0
Observation Domain Id = 1

FlowSet Id = 02 (Data Template)
FlowSet Length = 100

Template Id = 257
Field Count = 18
Field No = 1 Len = 8    Type = 1 (BYTES)
Field No = 2 Len = 8    Type = 2 (PACKETS)
Field No = 3 Len = 1    Type = 4 (PROTOCOL)
Field No = 4 Len = 4    Type = 10 (INPUT_SNMP)
Field No = 5 Len = 16  Type = 27 (IPV6_SRC_ADDR)
Field No = 6 Len = 16  Type = 28 (IPV6_DST_ADDR)
Field No = 7 Len = 4    Type = 34 (SAMPLING_INT)
Field No = 8 Len = 1    Type = 35 (SAMPLING_ALG)
Field No = 9 Len = 6    Type = 56 (SRC_MAC)
Field No = 10 Len = 6  Type = 57 (DST_MAC)
Field No = 11 Len = 1  Type = 60 (IP_PROTOCOL_VERSION)
Field No = 12 Len = 8  Type = 152 (flowStartMillisec)
Field No = 13 Len = 8  Type = 153 (flowEndMillisec)
Field No = 14 Len = 2  Type = 410 (GOOSE_APPID)
Field No = 15 Len = 64 Type = 411 (GOOSE_CB_REF)
Field No = 16 Len = 64 Type = 412 (GOOSE_DATA_SET)
Field No = 17 Len = 64 Type = 413 (GOOSE_ID)
Field No = 18 Len = 4  Type = 414 (GOOSE_ST_NUM)
```

(b) IPFIX GOOSE Flow Record

```
Version = 10
Length = 592
Timestamp = Nov 03, 2017 14:30:18
FlowSequence = 0
Observation Domain Id = 1

FlowSet Id = 257 (Data)
FlowSet Length = 576

Flow 1

8520
60
0 = IPv6 Hop-by-Hop Option
0
fe80::209:8eff:fefb:4c2b
ff02::10c:cd01:0:2
0
0 (No sampling)
00:09:8e:fb:4c:2b
01:0c:cd:01:00:02
6 = IPv6
Nov 03, 2017 14:29:27.232
Nov 03, 2017 14:30:25.282
0x01
SCU1UD6/LLN0$GO$Control_Dataset
SCU1UD6/LLN0$Dataset
SCU1/UD6/LLN0/Control_Dataset
1
```

Figure B.7: A GOOSE template record (a) and the flow record example (b)

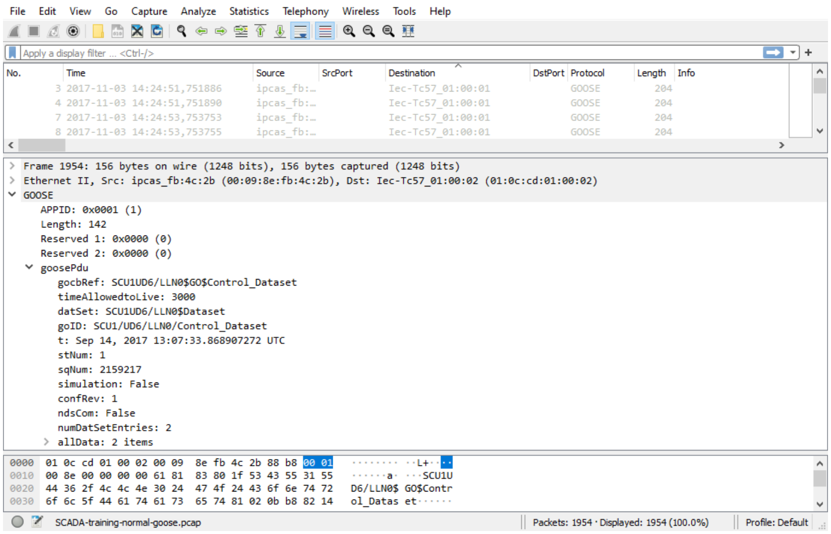
The template contains standard IPFIX fields like the number of transmitted packets and bytes, IP protocol, sampling time and algorithm,the start and the end of flow. It also includes standardized field source and destination MAC address, and source and destination IPv6 address. GOOSE specific fields include Application ID, GOOSE control block, Data Set name, GOOSE ID, and the Status Number.

Looking at Figure B.7 (b), you can notice that the example GOOSE flow transmits 60 packets of total size 8520 bytes, the flow took about 30 seconds and it was sent by publisher at 00:09:8e:fb:45c:2b to subscribers listening on the multicast address 01:0c:cd:01:00:02. Packet transmission was controlled by GOOSE Control Block *SCU1UD6/LLN0$GO$Control_Dataset*, contained GOOSE ID *SCU1/UD6/LLN0/Control_Dataset* and the name of the Data Set was *SCU1 UD6/LLN0$Dataset*.

For security monitoring it is important to understand, how many GOOSE packets are usually exchanged within a group *publisher-subscriber*. Also, for whitelisting it is useful to filter unknown source and destination MAC addresses, GOOSE

Application IDs, GOOSE Control Blocks, Data Set names, or GOOSE IDs. Since GOOSE communication is highly regular, it is natural to apply statistical methods to observe GOOSE behavior as presented in Chapter 5.

An example of GOOSE packet analysis in Wireshark and an IPFIX flow record with GOOSE fields is given in Figures B.8 and B.9.



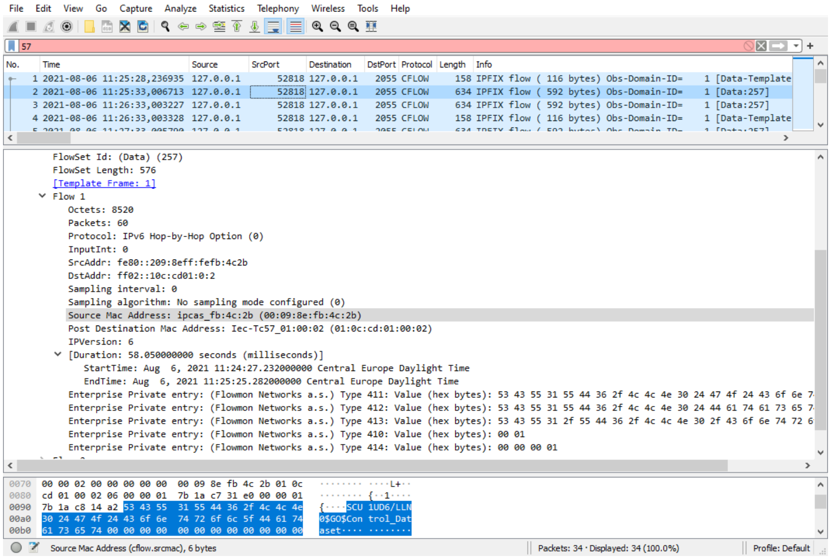
```
No.    Time                        Source       SrcPort  Destination           DstPort Protocol  Length  Info
     3 2017-11-03 14:24:51,751886  ipcas_fb:…            Iec-Tc57_01:00:01             GOOSE     204
     4 2017-11-03 14:24:51,751890  ipcas_fb:…            Iec-Tc57_01:00:01             GOOSE     204
     7 2017-11-03 14:24:53,753753  ipcas_fb:…            Iec-Tc57_01:00:01             GOOSE     204
     8 2017-11-03 14:24:53,753755  ipcas_fb:…            Iec-Tc57_01:00:01             GOOSE     204
```

```
> Frame 1954: 156 bytes on wire (1248 bits), 156 bytes captured (1248 bits)
> Ethernet II, Src: ipcas_fb:4c:2b (00:09:8e:fb:4c:2b), Dst: Iec-Tc57_01:00:02 (01:0c:cd:01:00:02)
v GOOSE
    APPID: 0x0001 (1)
    Length: 142
    Reserved 1: 0x0000 (0)
    Reserved 2: 0x0000 (0)
  v goosePdu
      gocbRef: SCU1UD6/LLN0$GO$Control_Dataset
      timeAllowedtoLive: 3000
      datSet: SCU1UD6/LLN0$Dataset
      goID: SCU1/UD6/LLN0/Control_Dataset
      t: Sep 14, 2017 13:07:33.868907272 UTC
      stNum: 1
      sqNum: 2159217
      simulation: False
      confRev: 1
      ndsCom: False
      numDatSetEntries: 2
    > allData: 2 items
```

```
0000  01 0c cd 01 00 02 00 09  8e fb 4c 2b 88 b8 00 01   ········ ··L+····
0010  00 8e 00 00 00 00 61 81  83 80 1f 53 43 55 31 55   ······a· ···SCU1U
0020  44 36 2f 4c 4c 4e 30 24  47 4f 24 43 6f 6e 74 72   D6/LLN0$ GO$Contr
0030  6f 6c 5f 44 61 74 61 73  65 74 81 02 0b b8 82 14   ol_Datas et······
```

SCADA-training-normal-goose.pcap || Packets: 1954 · Displayed: 1954 (100.0%) || Profile: Default

Figure B.8: Analysis of an GOOSE packet in Wireshark



```
No.  Time                        Source     SrcPort Destination DstPort Protocol Length Info
   1 2021-08-06 11:25:28,236935  127.0.0.1    52818 127.0.0.1      2055 CFLOW       158 IPFIX flow ( 116 bytes) Obs-Domain-ID=     1 [Data-Template
   2 2021-08-06 11:25:33,006713  127.0.0.1    52818 127.0.0.1      2055 CFLOW       634 IPFIX flow ( 592 bytes) Obs-Domain-ID=     1 [Data:257]
   3 2021-08-06 11:26:33,003227  127.0.0.1    52818 127.0.0.1      2055 CFLOW       634 IPFIX flow ( 592 bytes) Obs-Domain-ID=     1 [Data:257]
   4 2021-08-06 11:26:33,003328  127.0.0.1    52818 127.0.0.1      2055 CFLOW       158 IPFIX flow ( 116 bytes) Obs-Domain-ID=     1 [Data-Template
```

```
      FlowSet Id: (Data) (257)
      FlowSet Length: 576
      [Template Frame: 1]
    v Flow 1
        Octets: 8520
        Packets: 60
        Protocol: IPv6 Hop-by-Hop Option (0)
        InputInt: 0
        SrcAddr: fe80::209:8eff:fefb:4c2b
        DstAddr: ff02::10c:cd01:0:2
        Sampling interval: 0
        Sampling algorithm: No sampling mode configured (0)
        Source Mac Address: ipcas_fb:4c:2b (00:09:8e:fb:4c:2b)
        Post Destination Mac Address: Iec-Tc57_01:00:02 (01:0c:cd:01:00:02)
        IPVersion: 6
      v [Duration: 58.050000000 seconds (milliseconds)]
            StartTime: Aug  6, 2021 11:24:27.232000000 Central Europe Daylight Time
            EndTime: Aug  6, 2021 11:25:25.282000000 Central Europe Daylight Time
        Enterprise Private entry: (Flowmon Networks a.s.) Type 411: Value (hex bytes): 53 43 55 31 55 44 36 2f 4c 4c 4e 30 24 47 4f 24 43 6f 6e 7
        Enterprise Private entry: (Flowmon Networks a.s.) Type 412: Value (hex bytes): 53 43 55 31 55 44 36 2f 4c 4c 4e 30 24 44 61 74 61 73 65 7
        Enterprise Private entry: (Flowmon Networks a.s.) Type 413: Value (hex bytes): 53 43 55 31 2f 55 44 36 2f 4c 4c 4e 30 2f 43 6f 6e 74 72 6
        Enterprise Private entry: (Flowmon Networks a.s.) Type 410: Value (hex bytes): 00 01
        Enterprise Private entry: (Flowmon Networks a.s.) Type 414: Value (hex bytes): 00 00 00 01
```

```
0070  00 00 02 00 00 00 00 00  00 09 8e fb 4c 2b 01 0c   ········ ····L+··
0080  cd 01 00 02 06 00 00 01  7b 1a c7 31 e0 00 00 01   ········ {··1····
0090  7b 1a c8 14 a2 53 43 55  31 55 44 36 2f 4c 4c 4e   {····SCU 1UD6/LLN
00a0  30 24 47 4f 24 43 6f 6e  74 72 6f 6c 5f 44 61 74   0$GO$Con trol_Dat
00b0  61 73 65 74 00 00 00 00  00 00 00 00 00 00 00 00   aset···· ········
```

Source Mac Address (cflow.srcmac), 6 bytes || Packets: 34 · Displayed: 34 (100.0%) || Profile: Default

Figure B.9: IPFIX record with the GOOSE fields in Wireshark