SECURE MULTICAST FOR POWER GRID COMMUNICATIONS

BY

JIANQING ZHANG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

Abstract

Secure multicast for power grid systems faces a number of challenges like complex and error-prone group configuration, inefficient group key management, real-time challenges to existing security protocols and the balance among correctness, efficiency, feasibility and cost.

We propose an application-aware approach to setting up secure multicast for power grid communications that automatically derives group memberships and verifies configuration conformance from data dependencies in system specifications. We present an analytic publish-subscribe model, which formally depicts the relationships between data objects, publishers, subscribers and group controllers in a secure multicast system. Based on the model, we study anomalies in multicast functionality configurations like redundant and unauthorized publications, source-anomaly and data-dissatisfaction subscriptions. Algorithms are developed to detect the anomalies and verify the configuration conformance. A practical architecture is designed for automatic and error-resistant group configuration. It transforms the application layer system specifications to the network layer group security associations, policies and credentials. We also demonstrate the feasibility of raising link layer control messages to the network layer and protecting timing critical multicast traffic using one of the off-the-shelf network layer security protocols, namely IPsec. We provide experimental evidence that native IPsec multicast is capable of addressing latency constraints in medium scale networks.

To evaluate the approach, we present a case study of IEC 61850 power substation networks and have developed a demo system, SecureSCL. The case study shows the benefits a real-world application gains from the automatically-generated group security configurations and demonstrates the practicality and efficiency of the approach.

This work provides a cross-layer approach of automatically self-generated group configuration for power grid communications, addressing key concerns of both system implementation and conformance analysis. The proposed multicast model and verification mechanism can be extended for generic secure communication configurations. On the other hand, the prototype system SecureSCL has a potential of being developed into a realistic application for power substations.

To my beloved wife Ying Wang

Acknowledgement

First and foremost, I would like to express my great appreciation to my supervisor, Professor Carl A. Gunter, for his guidance, assistance, encouragement during this research and many years Ph.D. study. The opportunities and learning experiences he has given me are deeply appreciated. My experience at University of Illinois at Urbana-Champaign and University of Pennsylvania is especially rewarding and helpful in my future career because of his patient and strong supports not only in the research work but also in the career design. I'm grateful for his unconditional moral support, which is one of the factors that led me to completion of my Ph.D. studies.

I would like to thank my thesis committee members. Professor Roy Campbell raised my attention to some problems I missed at the beginning. Professor William Sanders not only offered good advice to the dissertation but also created a great environment in the TCIP group, which is a strong backup for my research. Professor Samuel King provided insightful comments for the system analysis of PCES-HS. Mr. Scott Mix helped me identify the real problems in power industry. Their efforts to my thesis work help me accomplish the Ph.D. study with solid output.

I have a great research group. Micheal LeMay was a wonderful office-mate for many years and I enjoyed our joint research efforts and interesting technical talks. The conversation to Alwyn Goodloe helped me thoroughly understand IPsec and related topics. I would also like to acknowledge the support and the friendship from others working in the group, in particular, Raja Afandi, Jodie Boyer, Sonia Jahid, Hee-Dong Jung, Fariba Khan, Lars Olson, Ravinder Shankesi and Kaijun Tan.

Thanks to the Information Trust Institutes TCIP project for awarding me a research assistantship, providing me with the financial means to complete this project. I would like thank Dr. Himanshu Khurana, Dr. Rakesh Bobba, Dong Jin and Tim Yardley. The conversation with them about secure multicast in power grid systems and performance measurement approaches inspired my research work. Thanks to Chris Grier, his work on PCES-HS was significant for my early work. I'm thankful for great help from Bruce Muschlitz. He is one of my most important sources for the knowledge about utility communications. He helped me understand how a power substation works and figure out practical and reasonable case studies.

Above all, I would like to give special thanks and appreciations to my beloved, wonderful wife, Ying Wang, for her love, patience, understandings, scarifies and encouragements during this research. I'm grateful for her enduring this long process with me. Without her always and firm support and love, especially at those difficult moments, this doctoral work would not have been possible.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

1.1.1 Multicast in Power Grid Communications

Power grids play a key role in national and economic security, public health, and safety. Its reliability affects society seriously. For example, on August 14, 2003, large portions of the Midwest and Northeast United States and Ontario, Canada, experienced a cascaded electric power blackout. The outage affected an area with an estimated 50 million people and 61,800 megawatts (MW) of electric load. The estimated total costs ranged between $4 billion and $10 billion US dollars in the United States, and $2.3 billion Canadian dollars in Canada [77].

To ensure the reliable and continuous supply of electricity, a variety of communication technologies are used to transmit critical information for power grid monitor and control. Multicast [17] is one of the mechanisms that are used widely in power grid communications. For example, UDP/IP multicast is used in some DNP3 [20] applications to reset counters and/or energy values of multiple remote control devices simultaneously at all locations, so that a definite synchronization point can be made. IP multicast is considered in Phasor Measurement Units (PMUs) [56] for delivering status data periodically in a large geographic area since it can cross network segments and uses bandwidth efficiently. In IEC 61850 [35] power substations, link layer multicast protocols, like Generic Object Oriented Substation Events (GOOSE) and Sampled Measured Value (SMV), are used to collect power grid real-time status, update the state of Intelligent Electric Devices (IEDs), and deliver control commands.

1.1.2 Security Requirements

Previously, power grids are separated from public networks by proprietary protocols and dedicated communication channels. The security of control devices relies on the physical isolation and network perimeters like firewalls, gateways or VPNs. However, as power grid communications are migrating from industry proprietary infrastructures to public infrastructures and protocols [84, 58, 41, 32, 73, 67], cyber security risks are also increased beyond those encountered when such systems rely on physical isolation for protection.

For example, within many power substation networks, wireless devices are used to collect power grid data from on field sensors. An adversary can sniff and collect critical data from the wireless network by compromising a wireless sensor or setting up a malicious device. Some engineers often use laptops to maintain or configure IEDs in a substation. If the laptops have access to the Internet via independent links like 3G networks, the firewall of the substation network would be bypassed and the laptop could become a back-door for hackers. Besides, the firewall or the gateway of the network could be compromised due to active attacks or misconfiguration [3, 26]. Therefore, the assumption that the internal substation network is isolated and secure is not true anymore. All these scenarios would lead to security risks to the substation network.

Therefore, the conventional security requirements in the Internet, like confidentiality, integrity and availability, are applied to power grid networks as well [78, 39, 51, 64, 48, 25, 62]. For example, some raw power grid data must be encrypted because they can be used to estimate the electricity price, which usually is the significant commercial secret for a power generation plant or a utility company. The communication system must prevent tampered status data or falsified control commands, which can lead to incorrect control decisions or actions. DoS attacks or data floods due to devices malfunction should be mitigated since they can overwhelm computerized sensors and actuators or communication networks [15]. To improve the efficiency of system maintenance or problem diagnosis of the whole power grid, it is very helpful to share data among power plants, utilities and regulators. However privacy will be a big concern since those organizations do not want to expose their customers' information or publicize the defects in their systems.

In summary, security, especially the integrity of multicast, will be one of the most interesting and challenging problems for power grid systems.

1.1.3 Challenges of Secure Multicast

There is an interest in providing security guarantees using cryptographically secured protocols [36, 22, 23, 52]. However, these solutions have been inadequate consideration of secure multicast. Some particular challenges must be addressed for secure multicast solutions of power grid.

Latency requirements Various application requirements [72] lead to latency challenges to existing security protocols. Some critical messages must be delivered within a threshold determined by power system functionalities. For example, GOOSE messages are usually required to be delivered between 2 and 10 milliseconds. PMU systems have transmission frequency requirements at 30 times per second or even higher. To protect the value of the power grid monitor and control, applications of security protocols must be done with as little impact on latencies as possible. Naive approaches to securing these messages with the required latency usually do not succeed. An enhanced secure multicast scheme would be necessary for timing-critical multicast communications.

Manageable configuration Because of intricate system designs, the need to integrate proprietary configuration tools from multiple vendors, and the complexity of configuring current off-the-shelf security protocols, it is a complex and error-prone task to configure group memberships, policy and keys for a large multicast system.

In a typical power grid multicast environment like GOOSE in IEC 61850 power substations, there are tens or hundreds of multicast groups. Each group member may appear in different groups with different roles, either a publisher or a subscriber. The payloads of messages in different groups differ from each other significantly. The configuration information of these groups and the message payloads, as well as the overall system design, is stored in a collection of large and complicated configuration files.

During the system design phase, the system requirements, functionalities on each IED and the substation network configuration change frequently. A single change to an individual device may affect a number of relevant devices and lead to corresponding changes in the other parts of the substation. Because the configuration files are usually edited or managed manually or by some basic tools without self-checking, it is very likely to miss accordingly updates when some parts of the system configuration are changed and cause anomalies or inconsistencies. Such configuration mechanisms are often inefficient and error-prone, just like what happen in firewalls and IPsec policy configuration [3, 26, 83]. At the same time, to reduce the risk of the system malfunction due to the design or quality defects in the IEDs from a particular manufacturer, utilities usually deploy control devices from multiple vendors in a substation. Engineers have to integrate multiple proprietary configuration tools from different vendors. This strategy makes the configuration more complicated and harder to audit. Furthermore, the complexity of configuring current off-the-shelf security protocols makes the problem more severe.

Functional configuration mistakes could lead to security violations [3, 26]. For example, according to incorrect group configuration, an IED could join a group where it should not appear, and deliver unnecessary data. This would violate the principle of least privilege.

Therefore, it is a big challenge to configure and manage secure multicast systems in power grids. An automatic, error-resistant and manageable configuration mechanism will improve the efficiency and mitigate inconsistency and mistakes in system design and deployment.

Efficient and feasible group key management Key management is always a big concern for the deployment of secure communication protocols. It becomes more critical and difficult for secure multicast systems since more members are involved and the group management is also challenging. Although researchers already proposed a number of sophisticated group key management protocols or schemes, most of them are not standardized and hard to integrate with power grid multicast systems smoothly. What's more, the configuration of the group key management protocols make the whole system more complicated. A feasible and integrated group key management scheme is required for power grid multicast systems.

The balance of the performance The balance between correctness, feasibility, efficiency and cost must be considered carefully. It is a good strategy to take advantage of suitably chosen and enhanced off-the-shelf security technologies that make the solution simple and feasible to implement and deploy functions at low costs and high assurance.

1.2 Approach

To provide a sophisticated secure multicast solution for power grid communications with the concern of above challenges, we propose an application-aware approach to setting up multicast groups using network layer security.

The basic idea is to derive group memberships and publication-subscription relationships based on data dependencies determined during system functional configuration. This is based on the observation that the data are the focus of a publication-subscription system and connect all group members in a multicast application. The data dependencies can be extracted from an appropriate extension of system domain-specific specifications. This approach can automatically figure out the multicast groups by integrating the network layer group management with the application layer functional configurations. The integration would also ease the deployment of security solutions by avoiding a time-consuming security configuration task.

Based on the derived group memberships, we try to detect inconsistent configurations automatically using a configuration verification tool. The results will help power engineers correct or revise the original system configuration or even facilitate the system design.

By extending configuration files with security related information, the group key management system can be integrated with the multicast system smoothly. The group key exchange protocol can be configured based on application logic.

To secure link layer multicast packets with off-the-shelf security protocols, we propose to raise the link layer multicast to the network layer and secure multicast traffic using IPsec. This change achieves quite a few benefits like the support of commercial IPsec implementations and the capability of wide area multicast for inter-substation communications.

1.3 Contribution

In this work, we propose a multicast formal data model and a publish-subscribe model, which depict the publication-subscription relationships. Based on the model, we classify a number of configuration anomalies in multicast systems and design the algorithms to detect the anomalies by analyzing the relationships between data objects, multicast publishers and subscribers. The multicast model and the anomaly detection mechanisms provide a method to analyze and verify the validity of multicast groups and publication-subscription configurations.

A multicast and group key management architecture based on the Group Domain of Interpretation (GDOI) [10] is designed and then used to set up group security associations based on the derived group memberships and the configuration verification results. We show that the challenges of multicast configuration and integrated group key management can be overcome by linking network layer secure multicast configuration to application-specific configuration of power substations.

To demonstrate this methodology we take IEC 61850 power substation networks as a case study and have developed a prototype system *SecureSCL*, which extracts multicast groups for GOOSE from high-level specifications such as extended Substation Configuration Language (SCL). SecureSCL transforms derived group information and security extensions to IPsec multicast configurations. We argue that it is appropriate to raise GOOSE to the network layer for IPsec protection because our experiments show that IPsec multicast is capable of addressing latency constraints in medium scale networks. This yields an automatically-generated security configuration that has acceptable and scalable impact on latencies, hence solving the problem of seamless low-latency security for GOOSE. This approach is validated by using it on a portion of the SCL specification of an experimental substation of the Tennessee Valley Authority (TVA).

1.4 Thesis Statement

For power substation automation an application-aware multicast, which derives group memberships and publication-subscription relationships from application logic and data dependencies, can set up network layer multicast groups efficiently, direct group key management and minimize configuration mistakes. IPsec based multicast is capable of addressing timing requirements for secure multicast in power grid systems.

1.5 Organization of the Thesis

The rest of the thesis is divided into seven chapters as follows. In Chapter 2, we review the background of power grid communication, power substation automation and configuration, IPsec multicast and group key management protocols. We discuss the related work in Chapter 3. A formal model depicting multicast applications in substation networks is presented in Chapter 4. The multicast configuration anomalies and the detection algorithms are discussed in Chapter 5. We show the implementation of the system and the case study of TVA Bradley Substation network in Chapter 6. We design an experiment system and test the performance of IPsec based multicast in Chapter 7. Chapter 8 concludes and discusses the future work.

Chapter 2

Background

We begin by supplying background on the key ideas we need for our study. These concern power grid communications, power substation automation, especially IEC 61850, and network layer secure multicast based on IPsec.

2.1 Power Grid Communications

2.1.1 SCADA Systems

An electrical power grid is a complex interconnected network for delivering electricity from suppliers to consumers using transmission and distribution networks across a large geographical area, as illustrated in Figure 2.1 (revised from [7]).

To ensure the reliable and continuous supply of electricity, Supervisory Control and Data Acquisition (SCADA) systems for Energy Management Systems (EMSs) are used for system monitoring, automation, protection and network management [21, 7, 57]. SCADA is a large-scale, distributed measurement and control system, typically used for data collection, archiving, analysis and control at the supervisory level. Apparently, real-time and uninterrupted communications are vital for the reliable operation of SCADA/EMS and power grid systems.

Figure 2.2 shows a typical architecture of a SCADA system (revised from [70, 7, 53]). It usually consists of the following components:

- Central supervisory system. A central supervisory system acquires process data from remote devices via Master Terminal Unit (MTU, *a.k.a.* master station), stores and analyzes collected data, monitors and processes events, and sends control commands to the process. The centralized supervisory system is usually located in a *control center*. In a large scale and complex SCADA system, more than one control center may be deployed.

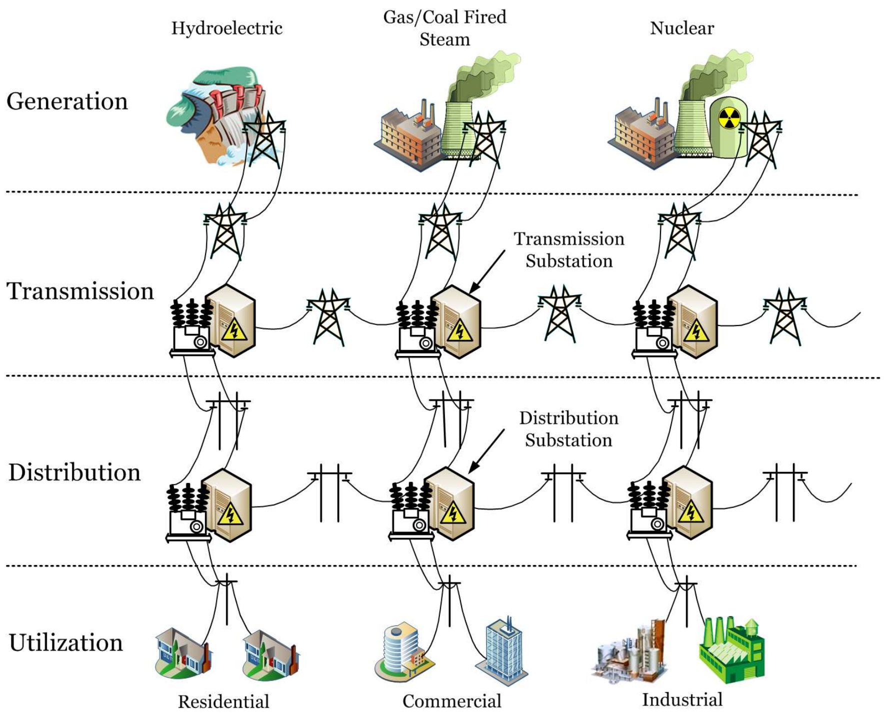


Figure 2.1: A Power Grid System

- Human-Machine Interface (HMI). HMI is the apparatus which presents process data to a human operator and through this the human operator monitors and controls the process. In addition to computer displays, HMI usually includes map boards, mimic diagrams or large group displays to provide an overview of system status.

- Remote Terminal Units (RTUs). An RTU is a microprocessor controlled electronic device which interfaces physical equipment or sensors in the process to a SCADA system. They convert electrical signals to digital values like the open/close status of a switch, or measurements like bus voltages and line currents, and transmit the telemetry data to the supervisory system. It can also control the equipment by converting and sending electrical control signals out to them. Sometimes RTUs work as data concentrators which get all process data from multiple physical devices into one place to make it easy for use with computers via communication protocols. RTUs and associated physical equipment are distributed at important areas of the power grid, like power substations and generation plants. In some cases, RTUs are substituted by Programmable Logic Controllers (PLCs). In addition to RTU's functionalities, PLCs support control algorithms or control loops. As hardware rapidly become more powerful and cheaper, RTUs and PLCs are increasingly beginning to overlap in responsibilities.

- Communication infrastructure. The communication infrastructure transmits information back and forth from the central supervisory system to the RTUs. Its physical media typically consists of serial links, leased lines, dedicated fiber, wireless (licensed microwave or unlicensed spread spectrum radio), satellite links, or even the Internet. The infrastructure also includes SCADA communication protocols. The legacy SCADA protocols are designed for low-bandwidth channels like serial links. They are very compact and many only send information to the master station when the master station polls the RTUs, like Modbus [54]. Recent protocols like Distributed Network Protocol (DNP3) [20] and IEC 61850 (see below) become much more sophisticated and many of them now contain extensions to operate over TCP/IP. Besides, web service and cloud computing technologies are also considered seriously by the power industry [41].

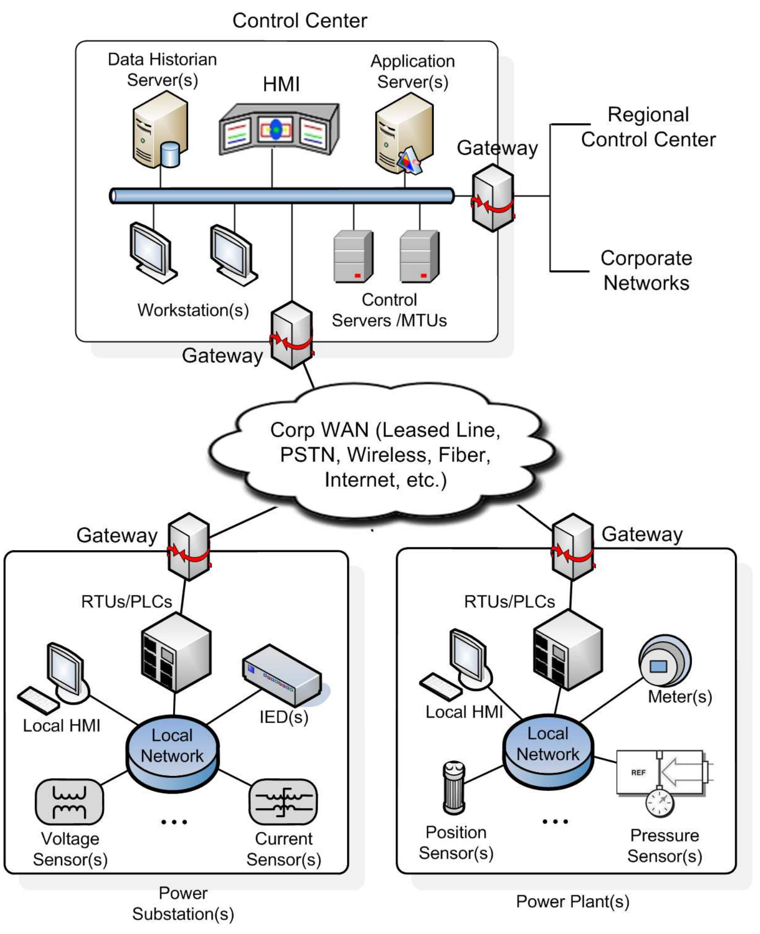


Figure 2.2: Architecture of a SCADA System

2.1.2 Data Types in Power Grid Communications

There are three basic types of data used in power grid communication: *protection system data*, *operational data* and *non-operational data* [53].

Protection system data are used for process control. They usually refer to control commands and critical system status updates, which have direct impacts on reliability and safety. Control commands are control devices' actions or responses based on the calculation of power grid status and system configurations. The system status data could be a feeder current value, a bus voltage value, a position indicator of a circuit breaker, *etc*. They are measured by sensors like relays or merge units (see Section 2.2.1) either as analog inputs by direct wires or digital samples by A/D conversion. Protection system data are strongly time essential with a requirement of response in milliseconds. Like most power grid communication applications, the transmission of protection system data is migrating from traditional autonomous point-to-point to packet-based network protocols like GOOSE. Because of the rigorous timing requirements, protection system data are not transferred to SCADA systems, *i.e.* they are usually limited within the local area network.

Operational data represent real-time status, performance and loading of power system equipment. They are required for SCADA systems to make supervisory decisions. Operational data are time critical but not as rigorous as protection system data. They are usually transmitted periodically and deterministically in seconds.

As the name suggests, non-operational data are not used for system operations like process control and supervisory control. They could be maintenance data, configuration information, revenue meter data, *etc*. Non-operational data are non-time critical and non-deterministic. They have no immediate or direct impact on reliability.

In this work, we focus on protection system data, especially the data transmitted in multicast.

2.2 Power Substation Automation

A power substation is a subsidiary station of an electricity generation, transmission, and distribution system. It is designed to control, monitor, and protect power grids. A substation usually consists of power system components like circuit breakers, transformers and switches, and control and monitoring components like RTUs, protective relays and meters. Nowadays, some components like protective relays are microprocessor-based and are often called Intelligent Electronic Devices (IEDs).

Substation Automation (SA) is an enhancement of traditional SCADA systems which rely on RTUs. SA takes advantage of the configurable communication IED technologies in implementing a local multi-level control hierarchy within a substation [57]. Logically there are two levels: a lower bay (or feeder) level and a higher substation level. It establishes a local area network between communicating IEDs and an SA gateway to manage the data within the substation. The gateway provides a communication interface back to the control center using SCADA protocols, and supports software-based internal substation interlocking and automation applications.

A substation network usually consists of tens or hundreds of IEDs. Previously, they were isolated from public networks by proprietary protocols and dedicated communication channels like leased lines, but this isolation is giving way to the benefits of broader and easier communication.

Nowadays, power grid communications are migrating from industry proprietary infrastructures to public infrastructures and open communication systems, often based on packet-based digital networking [84, 58, 41, 32, 73, 67]. IEDs are typically connected by Ethernet, TCP/IP and other protocols for exchanging power grid status information, delivering control commands, and setting configuration and/or maintenance parameters. Thin clients, web portals, and web based products are also gaining popularity with many vendors. This approach aids interoperability, visibility, and efficiency of system management and even has physical benefits like reducing the need for complicated wiring of serial links between control devices like IEDs.

Based on different physical interfaces and communication functionalities, a number of substation automation standards, like Modbus [54] and DNP3 [20] are designed. IEC 61850 [35] is one of the most recent, sophisticated and potentially prevailing specifications.

2.2.1 Overview of IEC 61850

IEC 61850 [35, 50, 66] supports a comprehensive set of substation functions and provides strong functional features for substation communications. It is easy for substation design, specification, configuration and maintenance. It is also extensible enough to support system evolution.

A typical IEC 61850 substation architecture is shown in Figure 2.3. There are several communication buses connecting all the IEDs inside a substation, which corresponds to SA levels. Substation buses, which are realized as medium bandwidth Ethernet networks, carry configuration and maintenance request/re-

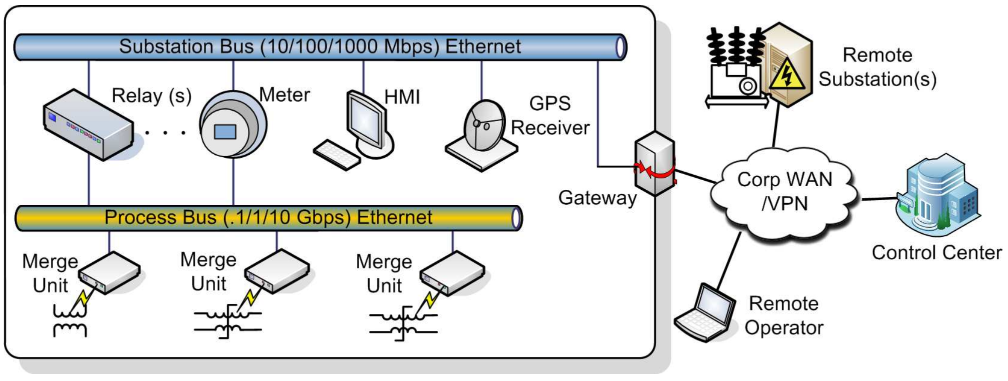


Figure 2.3: Architecture of an IEC 61850 Substation

sponses like Abstract Communication Service Interface (ACSI) messages, and Generic Substation Events (GSE) messages between IEDs and HMIs. Process buses based on high bandwidth Ethernet networks are designed for collecting real-time power grid status data. Merge units are deployed to digitalize analog power grid data, such as voltage and current, and multicast the sampled data to IEDs using Sampled Measured Value (SMV) messages. GPS is deployed for the time synchronization of the whole substation network. The devices within a substation can communicate with control centers, remote substations and remote operators via a gateway.

IEC 61850 consists of three major parts:

- An object data model describing the information available from different primary equipment types and from substation automation functions.
- A specification of the communication interfaces between IEDs and the schemes mapping them to a number of protocols running over TCP/IP and high speed Ethernet.
- An XML based configuration language used for exchanging the power system, substation network and devices configuration information.

In the rest of this section, we will introduce the three parts briefly.

2.2.2 Data Object Model

The IEC 61850 data model is designed to present and manage visible and accessible data within an IED (see Figure 2.4).

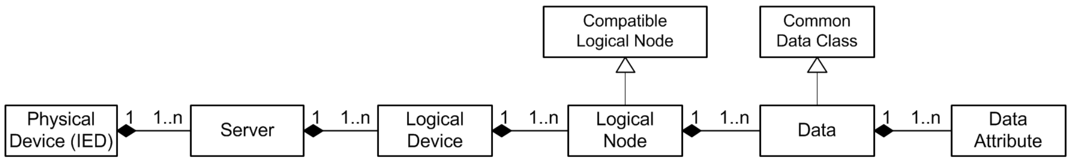


Figure 2.4: IEC 61850 Data Model

- *Server*: A server serves as a communication entity within an IED. It contains all data and services that are visible and accessible from the communication network. A physical device (IED) may host one or more server instances, and each server is bound with an *access point*, which is the logical representation of an IED's network interface.

- *Logical device*: A logical device (LD) is a group of domain specific application functions (*i.e.* logical nodes) and additional services. The grouping of logical nodes is based on their common features. In terms of the substation communication, a logical device serves as a unit for data services.

- *Logical node*: A logical node (LN) is a primitive, atomic functional building block in an IED. It is a named grouping of data and associated services that are logically related to a particular power function or application. IEC 61850 already defines a number of compatible logical node classes for well-known substation functions. For example, the class XCBR is defined to represent a basic circuit breaker. A practical logical node is a specialization of such a class and inherits the common features.

- *Data object*: Data exchanged between logical nodes are modeled as data objects. A data object represents a substation parameter, including its status, value and meta-information. For example, the data object Pos is often used to indicate the position of a switch ("on", "off", "intermediate-state" or "bad-state"). It also tells the meta-information like the timestamp of the status value and its originator. IEC 61850 defines a number of common data classes (CDC) as the templates for data objects. A CDC defines the whole set of data attributes necessary for a class of substation parameters. A data object defined in a compatible logical node class is a subtype of the corresponding CDC. For example, the data object Pos is a subtype of the class DPC (controllable double point).

- *Data attribute*: A data object consists of many data attributes. Actually data attributes are de facto logical correspondences to the physical values. For example, the data attribute stVal in Pos is the exact indicator of the position of the switch. Data attributes are typed and restricted by functional constraints (FC), which indicate which services can be used to access the values of the data attributes.

Besides that, the concept of *data set* is provided to manage and exchange a group of data attributes, which may belong to different data objects or logical nodes. Data sets are usually used to specify payloads of GOOSE or other messages.

2.2.3 Substation Communications

Abstract Communication Service Interface (ACSI) requests/responses, sampled analog values and GSE messages are three major kinds of data exchanged in IEC 61850 substation networks.

ACSI is designed for none timing-critical and client-server style message transmissions, including device configuration, maintenance, event logging and reporting. IEC 61850 defines a sophisticated communication profile to map abstract ASCI services to Manufacturing Message Specification (MMS) [44] and TCP/IP protocols through Open Systems Interconnection (OSI) [40] communication model stacks.

Sampled Measured Value (SMV) is an Ethernet link layer protocol used to periodically collect digitalized analog power data on the process bus. In most real deployments, however, the process bus and SMV are not implemented and IEDs still collect raw power data through analog signals via wires.

Generic Substation Event (GSE) is designed for fast and reliable system-wide distribution of input and output data values. It has two major forms: Generic Object Oriented Substation Event (GOOSE) and Generic Substation State Event (GSSE). GOOSE is used for fast exchange of a wide range of common data or substation events organized in a data set. GSSE provides backward compatibility with UCA 2.0 [38], the predecessor of IEC 61850. It just supports device state changes by fixed structures of the data in bit pairs. Both GOOSE and GSSE work in publisher-subscriber style and messages are transmitted by multicast.

In this work, we focus on GOOSE because it is based on the IEC 61850 data model and is the major mechanism for fast multicast communication in substations. GOOSE is also an Ethernet link layer multicast protocol designed for timing-critical messages within substation networks via substation buses. It is used for transmitting substation events, commands and alarms, *etc*. Because GOOSE is directly mapped to Ethernet frames, it can take advantage of high speed switched Ethernet and is capable of fulfilling real-time requirements.

In a typical scenario, to prevent a fault from being propagated, a protective relay multicasts one or more circuit breakers a TRIP command to disconnect the circuits upon detecting the fault. Figure 2.5 illustrates an

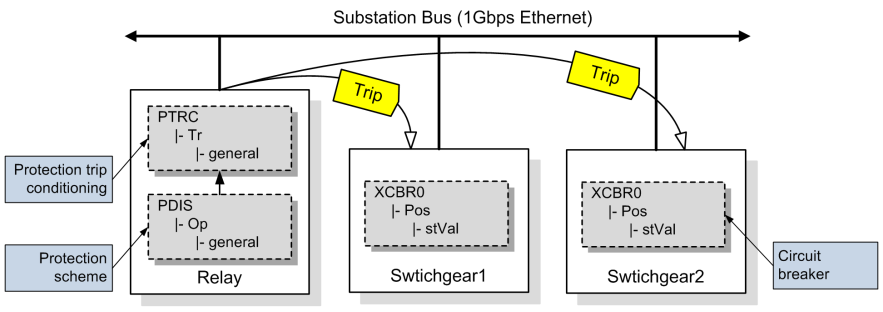


Figure 2.5: Timing-critical GOOSE Message in Power Substation

example where GOOSE is used to multicast a TRIP command from a distance protective relay to two circuit breakers. A distance relay starts and trips when the circuit admittance, impedance, or reactance increases or decreases (by measuring voltage and current) beyond a predetermined value. The logical node PDIS within the device represents the distance protection scheme by the data object Op. The logical node PTRC represents protection trip conditioning by the data object Tr. It is used to connect the "operate" outputs of one or more protection functions to a common "trip" to be transmitted to XCBRs. In this case, the relevant attributes of these two data objects like general are major parts of the TRIP command.

According to IEEE 1646 [72], event notification exchanges for protection within a substation must be transmitted within between 2 and 10 milliseconds. It is common to quote a benchmark of 4 milliseconds for this threshold so we use that figure in this work. The 4ms threshold is easily and reliably met by Ethernet multicast on commodity hardware at the load levels seen in power substations.

2.3 Substation Configuration

The basic purpose of substation configuration is to figure out a solution where the desired power functionalities of a substation can be realized by capable IEDs and associated equipment, and ensure the IEDs are appropriately configured and connected. It needs to specify power functions within the substation, describe the capabilities and customized parameters of IEDs and associated equipment, and depict the substation network topology, as well as the data flows between IEDs.

2.3.1 Substation Configuration Language

IEC 61850 defines XML-based Substation Configuration Language (SCL) for inter-operable exchange of communication system configuration data between different vendors and different configuration tools. It is a uniform description of the substation, and the relations between the substation and the SA functions, *i.e.* logical nodes in the IEDs. Based on the IEC 61850 data model, SCL defines an object model describing the IEDs, the substation network and their communication connections in terms of both application logic and network interfaces. From an SCL specification file we can obtain all information about the substation network topology, communication protocols, peer associations, and payload contents.

2.3.2 SCL Object Model

A simplified SCL UML object model is shown in Fig 2.6, which is a correspondence to the IEC 61850 data model. A typical SCL file consists of five types of elements: Header, Substation, Communication, IED and

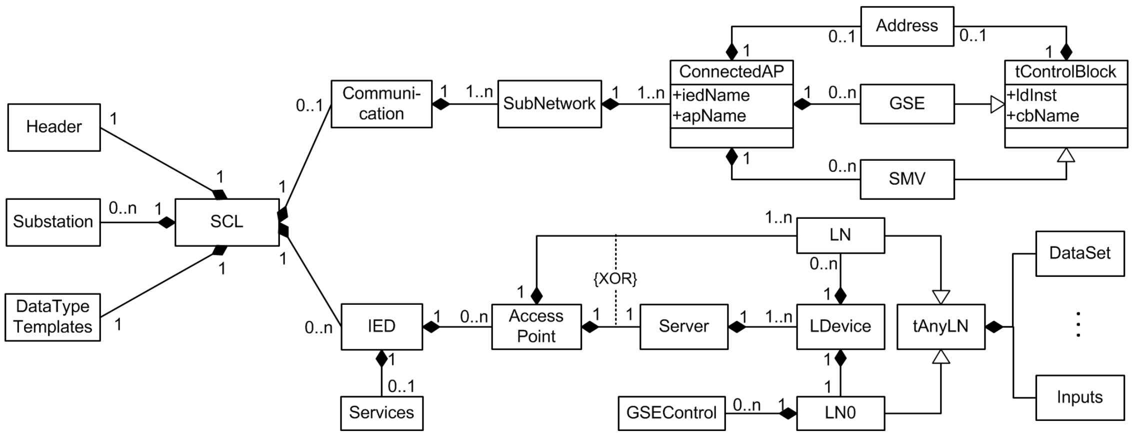


Figure 2.6: Simplified SCL UML Model

DataTypeTemplates[1]. The Header element is used to identify an SCL configuration file and its version. The Substation element(s) describe the functional structure of a substation, and identify the primary devices and their electrical connections. The DataTypeTemplate element defines instantiable logical node types, DATA types, structured attribute types and user-defined enumeration types. In this work, we focus more on the Communication and the IED elements.

The Communication element contains all information about the logically possible connections between IEDs at and across substation networks by means of access points. It consists of one or more SubNetwork elements, which are connecting nodes for direct (link layer) communication channels between access points. That also means a substation may have multiple LANs for substation buses and process buses. A logical device or a client of an IED is connected to a subnetwork by means of an access point, which may be a

[1]For simplicity, the terms of *element*, *object* and *section* are exchangeable in the rest of the section. All of them refer to an object in SCL.

physical port or a logical address (server) of the IED (see details below). A SubNetwork element consists of a number of ConnectedAP elements which represent the IED access points connected to this subnetwork. The attributes of iedName and apName identify the IED and the access point which is described in the corresponding IED element. ConnectedAP usually contains the address parameters of the access point at this bus via the Address element. If the access point serves ACSI, the Address element provides information including TCP/IP and OSI stack. If the access point also serves as a GSE server, *i.e.* a GSE publisher, the GSE element, a subclass of ControlBlock, is used to provide link layer network information like multicast MAC addresses. Each GSE element corresponds to one GSE application and the attributes of ldInst and cbName are used to identify the logical device which hosts the application, and the relevant control block. The mechanism also applies to SMV applications.

The IED element describes the pre-configuration of an IED: its access points, the logical devices, and logical nodes instantiated on it. Furthermore, it defines the capabilities of an IED in terms of communication services offered and, together with its logical node types, instantiated data and its default configuration values. There is one IED element for each IED in the substation. The Services element defines the available services, such as GOOSE, and their features on the IED.

An access point is a communication interface of an IED's logical devices to a substation network. A logical device usually has at most one connection, *i.e.* one access point, to a substation network, and multiple logical devices may share a single access point. The logical nodes contained in a logical device may use several access points as clients to connect to different subnetworks. A (logical) access point may support different physical network ports. For example, an Ethernet connection and a serial PPP based connection to the same higher level (TCP/IP) access point and to the same server. In SCL, the concept of access point is represented by an AccessPoint element, which consists of either a Server element or a number of LNs. If the access point is described as a server with logical devices, it provides access to the logical devices and logical nodes as data services. If the access point is described as a list of logical nodes, then it is used by the logical nodes as a client to get data from a process bus. In this work, we focus on the scenario where the access point serves as a data service.

The most important element within an IED server is LDevice. The LDevice defines a logical device of the IED accessible via the access point. An LDevice element contains at least one LN0, *a.k.a.* logical node zero, which represents common data and features of a logical device. The LN0 contains a number of elements which describe the control blocks for various communication applications. For example, the GSEControl identifies the name and the ID of a GSE application, and the name of the data set which is published by the application. An LDevice also intuitively contains a number of logical nodes represented by the elements of LN.

As the subtypes of tAnyLN, both LN0 and LN contain DataSet and Input. The DataSet represents a collection of data attributes of particular data objects, which are the message payloads of a GOOSE message or a reporting event. For example, Figure 2.7 shows the partial TRIP command illustrated in Section 2.2.3 in

```
<DataSet name="dsTripLogic">
    <FCDA daName="general" doName="Tr" fc="ST" ldInst="PROT" lnClass="PTRC" lnInst="1"/>
    <FCDA daName="t" doName="Tr" fc="ST" ldInst="PROT" lnClass="PTRC" lnInst="1"/>
    <FCDA daName="general" doName="Op" fc="ST" ldInst="PROT" lnClass="PDIS" lnInst="1"/>
    <FCDA daName="t" doName="Op" fc="ST" ldInst="PROT" lnClass="PDIS" lnInst="1"/>
    ...
</DataSet>
```

Figure 2.7: TRIP Command in Substation Configuration Language

SCL. The DateSet element consists of a number of FCDA (functionally constrained data attribute) elements. In this case all data attributes indicate the *status* of the logical nodes (by fc="ST") and usually read-only. Each FCDA is a reference to a relevant data attribute of the data objects Tr and Op in the logical node PTRC1 and PDIS1 (contained in the logical device PROT). Besides the general data attributes, the timestamps of each value are also specified by the data attribute t.

The Inputs element provides a references list to all data objects and their associated data attributes which are required by the logical node. Each reference indicates a data attribute by its naming and logical location information. The Inputs element is a good hint to figure out the publish-subscribe relationships between IEDs.

SCL defines how the configuration information is represented in files to be exchanged between engineering tools. It consists of four types of configuration files:

- *System Specification Description* (SSD). An SSD file describes the single line diagram of a substation and the allocation of logical nodes, *i.e.* required power functions.
- *IED Capability Description* (ICD). An ICD file describes the capabilities of an IED, *i.e.* the power functions (logical devices and logical nodes) and the pre-configured services the IED can provide. ICD files can be considered as "IED templates". Customization is needed during the configuration

process.

- *Substation Configuration Description* (SCD). An SCD file contains all information about the substation, including the functional structure of a substation, the primary devices and their electrical connections, all instantiated IEDs and the communication network configuration.
- *Configured IED Description* (CID). A CID file describes an instantiated IED within a project, including the network interfaces of the IED, instantiated logical devices and logical nodes. It is possibly a stripped-down SCD file to what the concerned IED shall know.

2.3.3 Configuration Process

A typical substation configuration process using SCL files is shown in Figure 2.8. Using a system specifica-

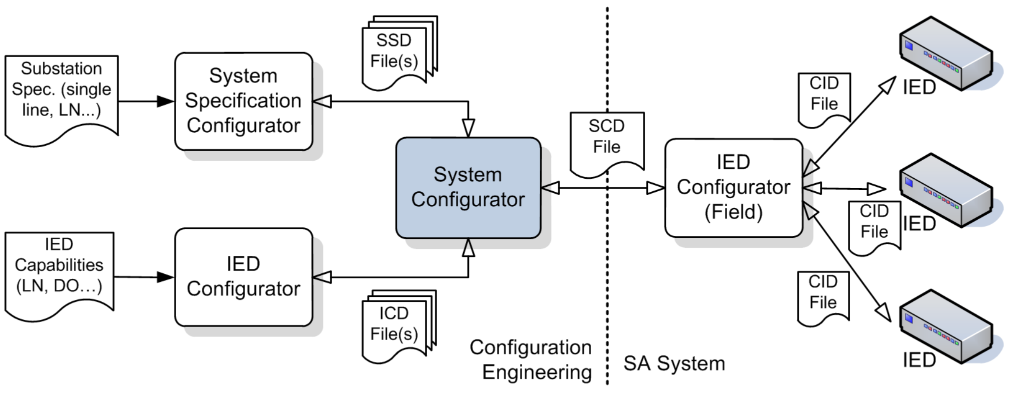


Figure 2.8: Substation Configuration Process

tion configurator, substation automation system (SAS) designers describe the needed data type templates and logical node type definitions in terms of the single line diagrams in SSD files. At this step, it is unnecessary to bind the needed logical nodes to particular IEDs. On the other hand, vendors usually use manufacturer-specific IED configurators to create ICD files. They provide the description of the pre-configured IEDs with a fixed number of logical nodes, available data services with pre-configured data sets, such as GOOSE and reporting. In most cases, logical nodes are only related to a very general process function part and no binding to a specific process. Some basic information like an IED's network interfaces or MAC addresses are presented in ICD files too. Importing the SSD and ICD files with a vendor-independent system configurator, SAS engineers complete process configuration with all IEDs bound to individual process functions and primary equipment. All logical devices, logical nodes and data objects are associated with real control processes. The IEDs are enhanced by the access point connections and possible access paths in substation networks for all possible clients, *i.e.* network interface configuration. Finally, a vendor-specific IED configurator working in the SA system reads the SCD file and loads each IED with all configuration parameters relevant to it. CID files are possibly created.

We should note that the substation configuration could be an iterative process. Interaction between vendors, SAS designers and engineers are required. The revisions or adjustments of the substation network and IEDs are inevitable. This is one of the reasons why configuration errors often occur.

2.4 Network Layer Secure Multicast

In 1990, Deering proposed IP multicast, an extension to the IP unicast service model for the efficient use of bandwidth for multi-point communication [17]. It uses the notion of a *group* of members associated with a given *group address*, *i.e.* a Class-D IP address. A sender simply sends a message to this group address and the network replicates the message at appropriate junctions like routers, and forwards the copies to group members throughout the network. Any entities, which are interested in the messages published in the group, can deliver the messages by listening to the group address.

There are a number of challenges for network layer multicast communications, including multicast routing and group membership management, *etc*. Security represents one of the major obstacles to the wide deployment of IP multicast. In [28], Hardjono and Tsudik defines three broad core problem areas for secure multicast, namely fast and efficient source authentication for high data-rate applications, secure and scalable group key management techniques and the need for methods to express and implement policies specific to multicast security.

In this work, we focus on an application-aware configuration approach which facilitates the deployment of multicast policies. We take advantage of off-the-shelf technologies based on IETF's efforts for secure IP multicast and relevant group key management protocols. The native IPsec and an IPsec based group key management solution are used to protect multicast traffic and distribute group keys in power substation networks.

2.4.1 IPsec Based Multicast Protocols

IPsec is originally designed as a pairwise security protocol for IP unicast communications. IPsec tunnels are set up by running the pairwise Internet Key Exchange (IKE) protocols [30, 45], which negotiate mutually acceptable Security Associations (SAs), like certificates and shared keys.

To protect IP broadcast/multicast applications within internal networks, Cisco proposes Generic Routing Encapsulation (GRE) and Dynamic Multipoint VPN (DMVPN), which support IP unicast, IP multicast and dynamic routing protocols (using GRE). Neither of them is a true IPsec multicast solution. They encapsulate multicast packets using IPsec tunnels between two gateways or routers, and the packets are only protected when they are transmitted through the tunnels. The multicast packets are payloads of the unicast IPsec packets. Behind the gateways, the packets are still in plain text, and no defense-in-depth is offered.

Actually, bulk IPsec implementations support multicast intrinsically [5, 11]. If an IPsec packet's destination address is a class-D address, *i.e.*, a multicast address, and the IPsec Security Policies (SPs) and SAs are configured properly, the IPsec packet can be received, decrypted and delivered to upper layer applications. Because most implementations enforce that each outgoing IPsec packet's source address must be same as the sender's IP address, individual source authentication is achieved in the network layer straightforwardly. To fully support wide area network multicast, extensions like Group Security Policy Database (GSPD) and multicast key update are needed [80].

In this work, we make use of native IPsec to protect timing critical multicast traffic between IEDs in a power substation network.

2.4.2 Group Domain of Interpretation

To support group key exchange, IETF designs the Group Domain of Interpretation (GDOI) [10], a multicast security and key management protocol based on [9] and [29]. Figure 2.9 shows the architecture of GDOI. A Group Controller & Key Server (GCKS) is introduced for group key management. On joining the group, new members authenticate themselves by setting up a Registration SA with the GCKS. Group members also pull Data SAs from GCKS for protecting multicast packets to and from other members. An optional Rekey SA is used to protect refreshing group keys which are generated and distributed by the GCKS or authorized members. The GDOI protocol borrows IKEv1 [30] Phase 1 for setting up Registration SAs and revises IKEv1 Phase 2 for distributing Rekey SAs and Data SAs. A working group from IETF is also working on

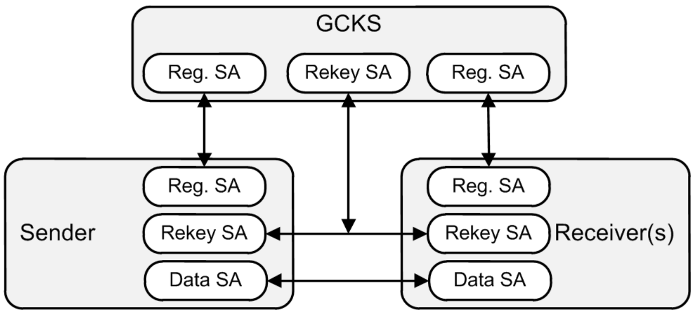


Figure 2.9: Architecture of GDOI

an revision of GDOI based on IKEv2 [45]. One of the major applications of GDOI is Cisco's DMVPN implementations, where GDOI is used to distribute group keys among gateways or routers for fast setting up dynamic VPNs. In this case, the GDOI protocol is only deployed on the gateways rather than individual hosts.

In this work, we make use of GDOI to achieve group key negotiations between individual hosts, specifically for substation IEDs.

Chapter 3

Related Work

In this chapter, we first introduce a couple of security protocols for link layer communications. Then we discuss some secure multicast schemes and group key schemes from both academic and industrial communities. We argue that standardized protocols with off-the-shelf implementations are appropriate for power grid multicast systems. In the end of the chapter, we introduce two projects about security configuration, which inspire the idea of the application-aware group derivation approach and the multicast model and anomaly detection algorithms in this work.

3.1 Link Layer Security Solutions

As mentioned in Section 2.2.3, both GOOSE and SMV are link layer multicast protocols. The original purpose for choosing the link layer for multicast is to achieve low latency performance. There are a number of link layer secure communication solutions. However, they are not capable of addressing the challenges to secure multicast in power grid systems.

IEC 62351 [36] is a specification suite for data and communication security of power grid systems. Its part 6 specifies message formats, procedures and algorithms for securing all protocols based on or derived from IEC 61850. It secures GOOSE and SMV by applying MAC (SHA256) and RSA signatures in the link layer. This solution requires significant computation latency for signing and verifying signatures on each frame. According to [46], in an ideal environment, it takes more than 2 milliseconds to sign and verify a packet using RSA 1024-bit keys. Considering the extra transmission overhead, this is risky to meet the 4ms threshold mentioned in Section 2.2.3. Indeed, IEC 62351-6 explicitly indicates that “this specification does define a mechanism for allowing confidentiality for applications where the 4ms delivery criterion is NOT a concern”. Furthermore, the communication overhead introduced by cipher text may cause the fragmentation issue which cannot be handled well by link layer protocols. The part 6 extends SCL to support certificates and secure access points, but no detailed implementation guide is presented.

IEEE 802.1AE [69] provides security for Ethernet frames using a hub-and-spokes topology. Security associations are set up between a switch and each connected host. The switch decrypts/encrypts frames, verifies/signs signatures of all relayed frames. Its default cipher suite is AES with 128-bit keys. IEEE 802.1AE introduces at least one more hop between the message sender and the recipient, which causes extra latency for message transmission (see detailed discussion about the hub-and-spokes topology in Section 7.1.2). IEEE802.1AE also requires hardware support so switches and NICs on all hosts must be upgraded. Therefore, it is not an appropriate solution for secure power grid multicast.

Like IEEE 802.1AE, Casado *et al.* propose two centralized network management systems for enterprise networks in [13] and [12]. They argue centralized network administration is acceptable because 1) enterprise networks are carefully engineered and centrally administered; 2) enterprise networks have predictable traffic. By modifying the link layer protocols, all traffic between two computers in an enterprise network will be authenticated and routed to a centralized controller which will forward the traffic to the destination. To reduce human errors in policy configuration, policy is defined and deployed centrally too. However, since their solution is intended for enterprise networks, some special constraints, such as real-time operation requirements, are not addressed. Like IEEE 802.1AE, there is at least one extra hop between the sender and the recipient. They have no application layer routing and access control. Tricky configuration in the application layer can violate the policy defined in centralized controllers.

3.2 Secure Multicast Schemes

Researchers have suggested a number of schemes for secure real-time multicast [82, 61, 59, 60, 79]. These schemes achieve the goals of integrity, fast-rate signature/verification and loss-tolerance by taking advantage of decent techniques like time synchronized MAC, authentication trees and reduced signature sizes *etc*. However, these advanced schemes are complicated to implement and few of them are standardized and/or commercialized. It is hard for industry to deploy them in real facilities.

In [27], Gjermundrod *et al.* propose GridStat, a publish-subscribe middle-ware system for power grid systems. They aim to address QoS on wide area networks. The messages transmitted in such an infrastructure are usually not protection system data and the timing constraints in the intended applications are not very crucial. Multicast configuration in local area network and group key management are also not their focuses.

Canetti *et al.* propose an IPsec-based host architecture for multicast in [11]. They introduce the concepts of Multicast Internet Key Exchange (MIKE) and Source Authentication Module (SAM), describe the functionalities of the architecture components and implement a prototype system for validation. Their work inspires our architecture in Section 6.2.2. They suggest implementing SAM in the application layer and designing the interfaces of SAM between the application layer and the network layer. Application data packets will be encapsulated by SAM before they are passed to IPsec modules. Our solution for individual source authentication is based on application logic but implemented in the network layer.

Aurisch *et al.* argue that IPsec can support secure multicast natively and present an implementation [5]. Weis *et al.* propose multicast extensions to IPsec to make it support wide area secure multicast better [80]. Our work focuses more on multicast group management and configuration. We also pay more attention on the performance issue for IPsec multicast and test the idea by experiments.

3.3 Group Key Management

Group key management is one of the most important components of secure multicast systems. Just like the research of secure real-time multicast, academic research also suggests a number of group key schemes [65, 14, 81, 6, 4]. These sophisticated solutions aim at the groups where group members churn frequently. The efficiency and scalability are the main concerns for these systems. However, in power grid systems, especially power substation networks, multicast groups are comparatively stable and the network scale is usually of medium size. Once the system design is finished, the network topology is rarely changed. On the contrary, it is hard to deploy the sophisticated schemes without commercial support.

In [29] and [9], two groups of researchers propose two very similar centralized group key management models. Both models introduce a group control and key server (GCKS) to manage group members and distribute/refresh group keys to group members. These two models are the bases for the GDOI [10]. Although both models discuss the (group) policy server and the authorization server, they are not implemented in the GDOI. In [31], Harney *et al.* propose a similar group key management framework based on IKEv2. Besides group key distribution, it provides more support for group membership management like the handshake process for joining or leaving a group. It also presents a trust and access control model for group communication.

In this work, we do not design a new group key management protocol. We take advantage of the GDOI and direct the group authorization and group policy configuration by application logic. We discuss the advantages of the GDOI for power grid multicast communication in Section 6.5.2.

3.4 Security Configuration

Ying *et al.* design an application-aware IPsec policy system in [85]. In the system, an application policy engine translates application policies into underlying proprietary security policies. A socket monitor is implemented to capture *socket()* calls, which are relevant to secured applications and match the security policies, write the security policies into IPsec SPD and invoke IPsec IKE. In contrast to their work, our architecture is based on standard configuration files and sets up IPsec SPD statically. Such solution makes a better use of existing configuration tools and requires less latency to protect application traffic. These features are more appropriate to power grid applications.

Al-Shaer *et al.* classify configuration anomalies in firewall filter policy configurations in [3]. They propose a model to present the anomalies formally and design algorithms based on state machines to detect the anomalies [2]. The firewall policy model and the anomaly detection algorithms partially inspire the multicast model in this work. We also try to classify multicast configuration anomalies and detect them.

Chapter 4

Multicast Modeling

In this chapter, we describe a formal model depicting multicast applications in substation networks. We begin with a simple example illustrating the type of applications we would like to model, and then represent a mathematical publish-subscribe model capable of precisely describing the relationships between the entities and the data in the example and others like it, including large practical specifications. Based on this model, we classify a number of multicast configuration anomalies, and develop analysis algorithms to verify the consistency of functionality and security configurations in Chapter 5.

4.1 Motivating Example

As an illustration let us consider an imaginary IEC 61850 power substation in which there are two protective relays P_1 and P_2, and four switchgears S_1, S_2, S_3 and S_4. Every IED has an id. According to the system design, each relay maintains two data objects Op and Tr, which are hosted in the logical nodes PDIS and PTRC respectively. Data objects on P_i are named Op_i and Tr_i for $i = 1, 2$, which actually represent the TRIP commands illustrated in Figure 2.5.

Additionally, to support particular remote collaborative functions, protective relays need to publish status information of other primary equipment, such as transformers, to circuit breakers periodically. In this example, each relay publishes an additional data set for this status information. The relay P_1 extends a class of general logical node, say GGIO (generic process I/O), by adding two data objects $St_{1,1}$ and $St_{1,2}$. The two data objects are mapped to two status parameters, like a feeder current or a bus voltage, and published on the substation bus. Similarly, P_2 publishes $St_{2,1}$, $St_{2,2}$ and $St_{2,3}$. Generally, the data objects $St_{i,j}$ are published by a relay P_i for $i = 1, 2$ and $1 \leq j \leq m_i$, where m_i is the number of status parameters published by P_i. In this example, $m_1 = 2$ and $m_2 = 3$.

In summary, on each relay P_i for $i = 1, 2$, two data sets $\{Op_i, Tr_i\}$ and $\{St_{i,j} : 1 \leq j \leq m_i\}$ are published in separate multicast groups with different multicast destination addresses.

Accordingly, to operate corresponding circuit breakers in case a fault occurs, S_1 and S_2 need monitor the data set $\{Op_1, Tr_1\}$ from P_1, while S_3 and S_4 need monitor the data set $\{Op_2, Tr_2\}$ from P_2. Furthermore, S_1 and S_3 also need monitor the status data set of $\{St_{1,1}, St_{1,2}\}$ from P_1, while S_2 and S_4 need monitor the data set $\{St_{2,1}, St_{2,2}, St_{2,3}\}$ from P_2. Each switchgear S_i has an additional data object Pos_i, which indicates the position of the circuit breaker (see Section 2.2.2). Based on application logic, circuit breakers are sometimes required to notify the relevant relays of the change of the switch position. For simplicity, the switchgears only update the value of Pos_i and do not establish extra multicast groups in this example. Fig 4.1 shows the illustrative diagram of the motivating example. The arrows show the multicast data flows

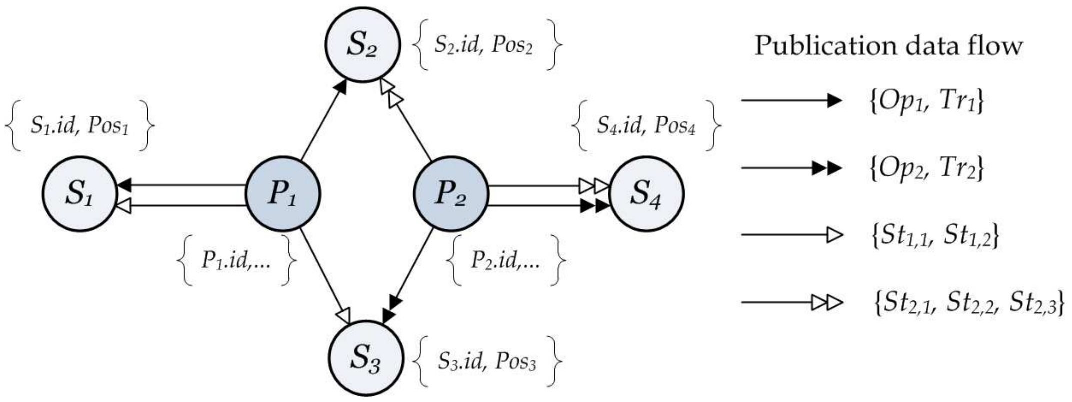


Figure 4.1: Motivating Example: Multicast in GOOSE Applications

and the payloads of each flow are specified on the right. The data objects that are not published are bracketed and showed close to entities.

All above design is specified in an SCD file. Each publication determines a multicast group with a unique multicast destination address, and the switchgears need join the corresponding groups. This network level configuration is also specified in the SCD file.

For simplicity, we do not use the sophisticated naming conventions defined in IEC 61850 and simplify the data model. The values of these data objects are the data attribute general in IEC 61850. Usually a real data set representing a TRIP command need more data attributes, like timestamps t.

We also simplify the transmission scheme used in GOOSE. Each GOOSE message has a state number and a sequence number. Messages are published repeatedly with increased time intervals rather than only once. If the payload has no any change, the sequence number will be increased by 1 and the state number will not be changed. The publisher will wait for longer time than last time to repeat multicasting the message.

If there is a change to the payload, *i.e.* a change to the data set, a new state number will be generated and the sequence number will be reset. A new message is sent immediately and the time interval for the next publishing is set to the smallest value. The subscribers can tell if the message is a fresh one or a duplication by checking the state number. The duplicated messages with the same state number will be discarded. By this scheme, GOOSE achieves message reliability in a simple and easy way. In our research work, we focus on the first message with the newly updated state number. The state number and the sequence number are not considered in the model. These simplifications aid our explanation but do not limit the applicability of our model in practice.

4.2 Components of a Secure Multicast System

A secure multicast system consists of a set of data objects, D, a set of data owners, O, a set of data consumers, C, a set of publishers, P, a set of subscribers, S, and a set of group controllers, G. Components which have relationships with data objects are called *entities*, E. Therefore, O, C, P and S are entities, *i.e.* $\mathsf{O}, \mathsf{C}, \mathsf{P}, \mathsf{S} \subseteq \mathsf{E}$.

4.2.1 Data Object

Data object is the core of the secure multicast model. In this work, our attention is focused on the *protection system data*, especially those data which are delivered using timing-critical multicast. As discussed in Section 2.1.2, they usually include physical parameters, environment conditions, control commands, *etc*.

Both control commands and critical system status data can be represented as a set of data objects. The design of the set depends on the application logic. In the motivating example, the data set $\{Op_1, Tr_1\}$ represents a TRIP command issued by P_1; the data set $\{St_{2,1}, St_{2,2}, St_{2,3}\}$ represents three critical power grid parameters measured by certain sensors and published by P_2.

In an IEC 61850 configuration file, the representation is a little bit complicated. The concept of data object in SCL is different from the one in our model. As introduced in Section 2.2.2, a data object in IEC 61850 represents a substation parameter, including its value and meta-information. It is made up of a number of data attributes, and the data attributes are the de facto logical correspondences to the physical values. Therefore, in our multicast model, when we talk about data objects, we actually refer to those data attributes. For example, Figure 2.7 shows a TRIP command represented in SCL. It consists of four data attributes from two IEC 61850 data objects. These four data attributes are mapped to four data objects in the multicast model.

4.2.2 Entity

As mentioned above, there are four types of entities in the multicast model: data owner, data consumer, publisher and subscriber. Each entity has a certain relationship with data objects, and the relationships are defined in configuration files.

As the name suggests, a *data owner* is an entity owns or hosts a number of data objects. Any control device with accessible data objects could be a data owner. It maintains the data objects by filling or updating the values based on configuration files or process monitoring. In the motivating example, the protective relay P_1 is the owner of the data object Op_1 and changes its value based on the calculation of real-time power grid status and pre-defined parameters. As mentioned above, an entity may own a variety of data objects and not all of them are published via a multicast group. That is, a data owner may not be a publisher in the multicast model. For example, S_1 owns data object Pos_1. But because it does not publish Pos_1, S_1 is not a publisher in this example.

A *data consumer* is an entity whose operations rely on certain data objects. It needs the data objects when the system is running. For example, the switchgear S_1 needs the data objects sets $\{Op_1.Tr_1\}$ and $\{St_{1,1}, St_{1,2}\}$. Like a data owner, a data consumer may require a variety of data objects and not all of them are delivered via multicast. For example, in IEC 61850 substations, an IED sometimes needs the data which are provided by the *report* mechanism. That is, the data consumer may not be a subscriber in the model.

A *publisher* is a content provider, which publishes data objects using multicast. It could be a protective relay which issues control commands, or a sensor which provides power status data to other devices. In this work, when we are talking about a publisher, it is always a sender in a multicast group. Apparently, an entity should only publish the data objects it owns, *i.e.* a publisher should be the data owner to the data objects it publishes. Unfortunately, due to configuration mistakes, a publisher defined in a configuration file could publish the data objects it does not own. One of this model's purposes is to help detect such mistakes.

A *subscriber* is an entity which subscribes to data objects from publishers. It could be a circuit breaker which executes commands issued by relays, or a protective relay which monitors power grid via the status update from sensors. Communication channels must be established between publishers and subscribers, *i.e.* a subscriber must be in the multicast group where the corresponding publisher sends the data objects. In this work, when we are talking about a subscriber, it is always one of recipients in a multicast group. Apparently, an entity should only subscribe the data objects it intends to consume, *i.e.* a subscriber should be the data consumer to the data objects it subscribes. However, due to configuration mistakes, a subscriber defined in a configuration file could subscribe to data objects it is not allowed to access.

Note that an entity could be either a publisher or a subscriber under different circumstances. For example, when a protective relay issues a TRIP command, it behaves as a publisher; when it monitors a circuit breaker's position status or collects raw data from merge units, it behaves as a subscriber. Without loss of generality, when modeling the multicast systems in substation networks and verifying the correctness of configurations, we study the multicast groups individually and the roles of publishers and subscribers in the group do not change.

4.2.3 Group Controller

A *group controller* provides group membership and group key management service. It is usually a piece of independent network equipment and does not host electrical data as entities do. It only exchanges group management and security related data with multicast groups. Security credentials in this model are not represented as data objects in this work.

A group controller performs the following tasks:

- Authorize group access privileges based on group memberships, which are derived from system configurations. The group controller accepts group join requests from certain entities by running group member authentication protocols. If an entity which should not in the group according to the system configuration but sends joining request, the group controller will cease the group authentication process and refuse to distribute group shared keys.

- Generate and distribute group keys. The group controller is responsible for refreshing and distributing group shared keys to group members.

- Revoke group memberships based on changing configurations. When engineers are configuring and changing parts of application logic, they may remove unnecessary entities from a group. The group controller is fed by updated configuration files and revokes removed group members by re-authenticating the whole group.

A real system may have multiple group controllers for the purpose of redundancy or fault tolerance. Without loss of generality, we only consider a single group controller in this work.

4.3 Publish-Subscribe Model

4.3.1 Assumptions

The publish-subscribe model describes the relations between entities and data objects in a multicast system. The purpose of a multicast application in power grid systems is to deliver certain data objects to a couple of indented recipients simultaneously and efficiently. In reality, a data object could be delivered to the recipients by different methods besides multicast publications. On the other hand, if an entity is interested in a data object, it is able to retrieve the data by different methods besides multicast subscriptions. To focus the attention to multicast applications in power grid communications, we have the following assumptions for the publish-subscribe model:

Assumption 4.1. *If a data owner publishes an owned data object and turns to be a publisher, it will not send the data object by other communication channels.*

It is possible for a data owner to provide a same data object to the IEDs in the substation using different communication methods. Besides publishing the data using multicast, it also can make the data accessible by providing client-server unicast protocols like MMS [44]. Since this work is focused on protection system data which are delivered by timing critical multicast communications, the data transmitted by other methods are out of scope of this work.

Assumption 4.2. *If a data consumer requires a data object, which is delivered in a publication, it only can deliver the data by subscribing to the publication.*

This is an extension of Assumption 4.1. If a publisher publishes a data object, the publication is the only way to access the required data objects for data consumers. Therefore, the data consumer only can subscribe the data object and turns to be a subscriber.

Assumption 4.3. *A data object is delivered by a publication exclusively, i.e. if a data object is delivered in a publication, it will not be delivered by another one.*

This assumption is a supplement to Assumption 4.2. It guarantees the subscription is *deterministic*, *i.e.* there is only one feasible publication, if a data consumer, as well as a subscriber, requires the data from a publication. In real applications, it is possible for a data owner or a publisher to put a same data object in two or more publications. This assumption will be waived in our future work.

4.3.2 Ownership

We define the *ownership* relation $\mathcal{R}_{own}$ as follows:

Definition 4.1. $\mathcal{R}_{own} \subseteq \mathsf{E} \times \mathsf{D}$: If an entity e *owns* or *hosts* a data object d, then $(e, d) \in \mathcal{R}_{own}$ or $\mathcal{R}_{own}(e, d)$, where $e \in \mathsf{E}$ and $d \in \mathsf{D}$.

For example, $\mathcal{R}_{own}(P_1.id, Op_1)$ and $\mathcal{R}_{own}(P_2.id, Tr_2)$. Apparently, if e owns a data object, then e is a data owner. Formally,

$$\mathcal{R}_{own}(e, d) \longrightarrow e \in \mathsf{O}, \tag{4.1}$$

where $e \in \mathsf{E}$ and $d \in \mathsf{D}$.

An entity usually owns a number of data objects, which are used in different applications or multicast publications. We define a function $\overline{\mathcal{R}_{own}}$, which takes an entity e as input and returns the set of data objects which are owned by e.

Definition 4.2. We define the function $\overline{\mathcal{R}_{own}} : \mathsf{E} \rightarrow 2^{\mathsf{D}}$ by the equation

$$\overline{\mathcal{R}_{own}}(e) = \{d \in \mathsf{D} : (e, d) \in \mathcal{R}_{own}\},$$

where $e \in \mathsf{E}$

For example, $\overline{\mathcal{R}_{own}}(P_1) = \{P_1.id, Op_1, Tr_1, St_{1,1}, St_{1,2}\}$ and $\overline{\mathcal{R}_{own}}(S_3) = \{S_3.id, Pos_3\}$. In a real system, an IED straightforwardly owns all data objects hosted on it.

4.3.3 Publication

We define the *publication* relation $\mathcal{R}_{pub}$ as follows:

Definition 4.3. $\mathcal{R}_{pub} \subseteq \mathsf{E} \times 2^{\mathsf{D}}$: If an entity e *publishes* a data set ds, then $(e, ds) \in \mathcal{R}_{pub}$ or $\mathcal{R}_{pub}(e, ds)$, where $e \in \mathsf{E}$ and $ds \in 2^{\mathsf{D}}$.

In the running example, $\mathcal{R}_{pub}(P_1, \{Op_1, Tr_1\})$ and $\mathcal{R}_{pub}(P_2, \{St_{2,1}, St_{2,2}, St_{2,3}\})$. Apparently, if an entity e publishes a data set, then it is a publisher. Formally,

$$\mathcal{R}_{pub}(e, ds) \longrightarrow e \in \mathsf{P}, \tag{4.2}$$

where $e \in \mathsf{E}$ and $ds \in 2^{\mathsf{D}}$.

So the *publication* relation can also be defined as $\mathcal{R}_{pub} \subseteq \mathsf{P} \times 2^{\mathsf{D}}$. Note that $\mathcal{R}_{own}$ and $\mathcal{R}_{pub}$ are two independent relations defined in configuration files. Although an entity only can publish the data objects it owns, an incorrect configuration may have it "publish" one or more data objects that are not owned by it. In this case, the entity is a publisher of the data objects but not an owner of them.

A publisher may have multiple publications. It is required to register individual multicast groups for each publication. We define a function $\widehat{\mathcal{R}_{pub}}$, which takes a publisher p as input and returns the union set of data sets which are published by p,

Definition 4.4. We define the function $\widehat{\mathcal{R}_{pub}} : \mathsf{P} \to 2^{2^{\mathsf{D}}}$ by the equation

$$\widehat{\mathcal{R}_{pub}}(p) = \{ds \in 2^{\mathsf{D}} : (p, ds) \in \mathcal{R}_{pub}\},$$

where $p \in \mathsf{P}$.

For example, $\widehat{\mathcal{R}_{pub}}(P_2) = \{\{Op_2, Tr_2\}, \{St_{2,1}, St_{2,2}, St_{2,3}\}\}$. Actually, for a given publisher $p \in \mathsf{P}$, the number of members of $\widehat{\mathcal{R}_{pub}}(p)$ implies the number of multicast groups that p supports. By deriving all these union sets from configuration files, we can get all multicast groups in a substation network and allocate network resources during the period of system design.

Further, we define a function $\overline{\mathcal{R}_{pub}}$, which takes a publisher p as input and returns the set of data objects which are published by p.

Definition 4.5. We define the function $\overline{\mathcal{R}_{pub}} : \mathsf{P} \to 2^{\mathsf{D}}$ by the equation

$$\overline{\mathcal{R}_{pub}}(p) = \bigcup \widehat{\mathcal{R}_{pub}}(p),$$

where $p \in \mathsf{P}$.

For example, $\overline{\mathcal{R}_{pub}}(P_2) = \{Op_2, Tr_2, St_{2,1}, St_{2,2}, St_{2,3}\}$. Given a publisher p, we can check if it has illegitimate publications by comparing the result of $\overline{\mathcal{R}_{pub}}(p)$ and $\overline{\mathcal{R}_{own}}(p)$. If $\overline{\mathcal{R}_{pub}}(p)$ is not a subset of $\overline{\mathcal{R}_{own}}(p)$, p publishes one or more data objects that do not belong to it. Please see Section 5.1 for details of publications anomalies in configuration.

4.3.4 Consumption

As discussed in Section 4.2.2, an entity sometimes needs get data input due to application logic. For a particular application, an entity usually requires more than one data object, ***i.e.*** a set of data objects is needed. So we define the *consumption* relation $\mathcal{R}_{con}$.

Definition 4.6. $\mathcal{R}_{con} \subseteq \mathsf{E} \times 2^{\mathsf{D}}$: if an entity $e \in \mathsf{E}$ *requires* a data object set, $ds \in 2^{\mathsf{D}}$, then $(e, ds) \in \mathcal{R}_{con}$ or $\mathcal{R}_{con}(e, ds)$.

For example, $\mathcal{R}_{con}(S_1, \{Op_1, Tr_1\})$ and $\mathcal{R}_{con}(S_3, \{St_{1,1}, St_{1,2}\})$. Apparently, if e consumes a data object set, and then e is a data consumer. Formally,

$$\mathcal{R}_{con}\ (e, d) \longrightarrow e \in \mathsf{C}, \tag{4.3}$$

where $e \in \mathsf{E}$ and $ds \in 2^{\mathsf{D}}$.

In fact, the consumption relation specifies an entity's access privileges of reading particular data objects. It represents the intended data flow between data owners and data consumers, and provides a method to enforce access control over data objects in a substation network. We will explore this topic further in Chapter 5.

An entity may require individual date object sets for each application or each function. For example, S_3 needs both $\{Op_2, Tr_2\}$ and $\{St_{1,1}, St_{1,2}\}$ for different purposes. We define the function $\widehat{\mathcal{R}_{con}}$, which takes a data consumer c as input and returns the union of data sets which are consumed by c.

Definition 4.7. We define the function $\widehat{\mathcal{R}_{con}} : \mathsf{C} \to 2^{2^{\mathsf{D}}}$ by the equation

$$\widehat{\mathcal{R}_{con}}(c) = \{ds \in 2^{\mathsf{D}} : (c, ds) \in \mathcal{R}_{con}\},$$

where $c \in \mathsf{C}$.

For example, $\widehat{\mathcal{R}_{con}}(S_3) = \{\{Op_2, Tr_2\}, \{St_{1,1}, St_{1,2}\}\}$.

Similarly, we define the function $\overline{\mathcal{R}_{con}}$, which takes a data consumer c as input and returns the set of data objects which are consumed by c.

Definition 4.8. We define the function $\overline{\mathcal{R}_{con}} : \mathsf{C} \rightarrow 2^{\mathsf{D}}$ by the equation

$$\overline{\mathcal{R}_{con}}(c) = \bigcup \widehat{\mathcal{R}_{con}}(c),$$

where $c \in \mathsf{C}$.

For example, $\overline{\mathcal{R}_{con}}(S_3) = \{Op_2, Tr_2, St_{1,1}, St_{1,2}\}$.

4.3.5 Subscription

Based on the above definitions, we define the *subscription* relation $\mathcal{R}_{sub}$ as a ternary relation.

Definition 4.9. $\mathcal{R}_{sub} \subseteq \mathsf{E} \times \mathsf{E} \times 2^{\mathsf{D}}$: if s *subscribes* to ds from p, then $(s, p, ds) \in \mathcal{R}_{sub}$ or $\mathcal{R}_{sub}(s, p, ds)$, where $s, p \in \mathsf{E}$ and $ds \in 2^{\mathsf{D}}$.

For a given $(s, p, ds) \in \mathcal{R}_{sub}$, the relation represents a subscription request sent by $s \in \mathsf{E}$, *i.e.* s is a subscriber. Formally, for a given $s \in \mathsf{E}$,

$$(s, p, ds) \in \mathcal{R}_{sub} \longrightarrow s \in \mathsf{S}, \tag{4.4}$$

where $s, p \in \mathsf{E}$ and $ds \in 2^{\mathsf{D}}$.

When such a subscription request is specified in a configuration file, it only represents the subscriber's data requirements in the system. It does not mean the request is legitimate and will be approved and authorized finally. First of all, s only can subscribe the data object which it has access to, *i.e.* the subscribed data set ds must be a subset of one of data sets s consumes. On the other hand, the second element of the relation $p \in \mathsf{E}$ should be a *valid* publisher. Furthermore, ds must be a subset of one of data sets p publishes, *i.e.* p does publish a data set which contains all data objects in ds. If all above requirements are satisfied, we call such a subscription *valid subscription*.

The formal description of valid subscription is defined below:

Definition 4.10. A subscription $(s, p, ds) \in \mathcal{R}_{sub}$ is *valid* if, and only if,

$$(p \in \mathsf{P}) \wedge [(\exists ds_p \in 2^{\mathsf{D}})(ds_p \subseteq \widehat{\mathcal{R}_{pub}}(p) \wedge ds \subseteq ds_p)] \wedge [(\exists ds_s \in 2^{\mathsf{D}})(ds_s \subseteq \widehat{\mathcal{R}_{con}}(s)) \wedge ds \subseteq ds_s)]$$

A subscription request can be considered as a group join request. It actually indicates which publication the subscriber is interested in, *i.e.* which multicast group the subscriber wants to join. Straightforwardly, the subscribers whose subscriptions refer to a same publication are the recipients of the multicast group.

4.4 Summary

In summary, we get the definition of the multicast mode.

Definition 4.11. A *multicast model*, $\mathcal{M}$ consists of:

- D, the set of data objects
- E, components which have relationships with data objects
- O, a set of data owners
- C, a set of data consumers
- P, the set of publishers
- S, the set of subscribers
- G, the set of group controllers

together with

- $\mathcal{R}_{own}$, the *ownership* relation: $\mathcal{R}_{own} \subseteq \mathsf{E} \times \mathsf{D}$
- $\mathcal{R}_{pub}$, the *publication* relation: $\mathcal{R}_{pub} \subseteq \mathsf{E} \times 2^{\mathsf{D}}$
- $\mathcal{R}_{con}$, the *consumption* relation: $\mathcal{R}_{con} \subseteq \mathsf{E} \times 2^{\mathsf{D}}$
- $\mathcal{R}_{sub}$, the *subscription* relation: $\mathcal{R}_{sub} \subseteq \mathsf{E} \times \mathsf{E} \times 2^{\mathsf{D}}$
- $\overline{\mathcal{R}_{own}}$, the function which returns set of the data objects owned by an entity: $\overline{\mathcal{R}_{own}} : \mathsf{E} \rightarrow 2^{\mathsf{D}}$

- $\widehat{\mathcal{R}_{pub}}$, the function which returns the union set of data sets, which are published by a publisher, $\widehat{\mathcal{R}_{pub}} : \mathsf{P} \to 2^{2^{\mathsf{D}}}$

- $\overline{\mathcal{R}_{pub}}$, the function which returns the set of data objects, which are published by a publisher, $\overline{\mathcal{R}_{pub}} : \mathsf{P} \to 2^{\mathsf{D}}$

- $\widehat{\mathcal{R}_{con}}$, the function which returns the union set of data sets, which are consumed by a consumer, $\widehat{\mathcal{R}_{con}} : \mathsf{C} \to 2^{2^{\mathsf{D}}}$

- $\overline{\mathcal{R}_{con}}$, the function which returns the set of data objects, which are consumed by a consumer, $\overline{\mathcal{R}_{con}} : \mathsf{C} \to 2^{\mathsf{D}}$

We depict the data model of the running example in Section 4.1 using $\mathcal{M}$ as following:

- $\mathsf{D} = \{P_1.id, P_2.id, S_1.id, S_2.id, S_3.id, S_4.id, Op_1, Op_2, Tr_1, Tr_2, St_{1,1}, St_{1,2}, St_{2,1}, St_{2,2}, St_{2,3}, Pos_1, Pos_2, Pos_3, Pos_4\}$

- $\mathsf{E} = \{P_1, P_2, S_1, S_2, S_3, S_4\}$

- $\mathsf{O} = \{P_1, P_2, S_1, S_2, S_3, S_4\}$

- $\mathsf{P} = \{P_1, P_2\}$

- $\mathsf{C} = \{S_1, S_2, S_3, S_4\}$

- $\mathsf{S} = \{S_1, S_2, S_3, S_4\}$

- $\mathsf{G} = \{KS\}$ (not shown in Figure 4.1)

- $\mathcal{R}_{own} = \{(P_1, P_1.id), (P_1, Op_1), (P_1, Tr_1), (P_1, St_{1,1}), (P_1, St_{1,2}), (P_2, P_2.id), (P_2, Op_2), (P_2, Tr_2), (P_2, St_{2,1}), (P_2, St_{2,2}), (P_2, St_{2,3}), (S_1, S_1.id), (S_1, Pos_1), (S_2, S_2.id), (S_2, Pos_2), (S_3, S_3.id), (S_3, Pos_3), (S_4, S_4.id), (S_4, Pos_4)\}$

- $\mathcal{R}_{pub} = \{(P_1, \{Op_1, Tr_1\}), (P_1, \{St_{1,1}, St_{1,2}\}), (P_2, \{Op_2, Tr_2\}), (P_2, \{St_{2,1}, St_{2,2}, St_{2,3}\})\}$

- $\mathcal{R}_{con} = \{(S_1, \{Op_1, Tr_1\}), (S_1, \{St_{1,1}, St_{1,2}\}), (S_2, \{Op_1, Tr_1\}), (S_2, \{St_{2,1}, St_{2,2}, St_{2,3}\}), (S_3, \{Op_2, Tr_2\}), (S_3, \{St_{1,1}, St_{1,2}\}), (S_4, \{Op_2, Tr_2\}), (S_4, \{St_{2,1}, St_{2,2}, St_{2,3}\})\}$

- $\mathcal{R}_{sub} = \{(S_1, P_1, \{Op_1, Tr_1\}), (S_1, P_1, \{St_{1,1}, St_{1,2}\}), (S_2, P_1, \{Op_1, Tr_1\}), (S_2, P_2,$ $\{St_{2,1}, St_{2,2}, St_{2,3}\}), (S_3, P_2, \{Op_2, Tr_2\}), (S_3, P_1, \{St_{1,1}, St_{1,2}\}), (S_4, P_2, \{Op_2, Tr_2\}),$ $(S_4, P_2, \{St_{2,1}, St_{2,2}, St_{2,3}\})\}$

In Chapter 5, we will use this model to formally depict the multicast configuration anomlies and design algorithms to detect the anomalies.

Chapter 5

Multicast Configuration Anomaly

As discussed in Chapter 1, it is a complex and error-prone task to configure group memberships, policy and keys for power grid multicast applications. During the process of power substation configuration, power engineers, utility communication engineers and vendors collaborate with each other for a comprehensive solution of the substation and the substation network. As introduced in Section 2.3, an SCD file will be generated, which contains all information about the substation, especially the information about all IEDs and the communication network. The SCD file is an integration of a number of ICD files and SSD files. It is subject to the change of the system requirements, the re-initialization of IEDs and the iterative design of the substation network. Because most of these changes are completed manually and there is no strong support from vendors' tools for conformance checking, the system configuration is subject to a variety of mistakes. These mistakes not only lead to system malfunctions but also introduce security vulnerabilities. It is a big challenge to guarantee the correctness and consistency of the configuration.

In this chapter, we discuss four types of anomalies in multicast configuration, especially those occurring in IEC 61850 power substation multicast applications. We first describe the anomalies with their reasons, occurrences and threats to the system. Then we represent the formal description of each anomaly based on the multicast model we propose in Chapter 4. We also show anomaly examples by adding additional specs to the motivating example in Section 4.1, which make it incorrect or inconsistent. We design algorithms to detect these anomalies in Section 5.2. Finally, we discuss the scope of the anomaly model and the algorithms.

5.1 Multicast Configuration Anomaly

There are two basic configuration settings for multicast applications: publications and subscriptions. We can find all the following configuration anomalies on either publisher's side or subscriber's side. We also can call them publication anomalies or subscription anomalies.

5.1.1 Ownership Anomaly

If a publisher p publishes a data set ds, which consists of the data objects that are not owned by p, we say the publication $\mathcal{R}_{pub}(p, ds)$ has an *ownership anomaly*. We give the formal definition below:

Definition 5.1. A publication $\mathcal{R}_{pub}(p, ds)$ has an *ownership anomaly* if

$$\exists d \in ds[d \notin \overline{\mathcal{R}_{own}}(p)]$$

or

$$\exists d \in ds[(p, d) \notin \mathcal{R}_{own}]$$

Generally, we say a publisher p has a *publication ownership anomaly* if it publishes data objects that not owned by it. A general definition is given below:

Definition 5.2. A publisher $p \in \mathsf{P}$ has a publication ownership anomaly if

$$\exists d \in \overline{\mathcal{R}_{pub}}(p)[d \notin \overline{\mathcal{R}_{own}}(p)].$$

For example, in the motivating example in Section 4.1, if a publication $\mathcal{R}_{pub}(P_1, \{Op_1, Tr_2\})$ was set up in the configuration file, the publication would have an ownership anomaly since Tr_2 is owned by P_2 rather than P_1.

In reality, if an IED is configured to take a data attribute, a data object, or even an entire data set, which is not owned by it, as parts or the whole payload of a GOOSE message, we say the IED has a publication ownership anomaly. Such anomaly usually occurs when the system data flow design is changed. Originally, the IED hosts one or more data objects or data attributes representing particular physical parameters and puts it in a publication. Due to the design change, the functions may be moved to another IED and the ownership of the data is changed. However, the publication configuration of the original IED does not change accordingly and the anomaly occurs.

5.1.2 Publication Redundancy

If a publisher p publishes a data set ds, but no entity consumes or subscribes to it, we say the publication $\mathcal{R}_{pub}(p, ds)$ is *redundant*. According to Assumption 4.2, if an entity consumes a data object, which is published in a publication, the entity subscribes to the publication as well. There are two types of publication redundancy: *full redundancy* and *partial redundancy*.

In the full redundancy, none of data objects in the data set are requested by any data consumers, *i.e.* the whole data set is redundant.

Definition 5.3. A publication $\mathcal{R}_{pub}(p, ds)$ is *fully redundant* if

$$\forall d \in ds\ \forall c \in \mathsf{C}\ [d \notin \overline{\mathcal{R}_{con}}(c)]$$

In the partial redundancy, some data objects in the data set are requested by some data consumers, while others are not.

Definition 5.4. A publication $\mathcal{R}_{pub}(p, ds)$ is *partially redundant* if

$$\exists d \in ds\ \exists c \in \mathsf{C}[d \in \overline{\mathcal{R}_{con}}(c)] \wedge \exists d' \in ds\ \forall c \in \mathsf{C}\ [d' \notin \overline{\mathcal{R}_{con}}(c)]$$

In the motivating example, a publication of $\mathcal{R}_{pub}(P_2, \{P_2.id\})$ would be a full redundant publication since no switchgear requests the relay's id. A publication of $\mathcal{R}_{pub}(P_2, \{P_2.id, Op_2, Tr_2\})$ would be a partially redundant publication since S_3 and S_4 only need Op_2 and Tr_2, and $P_2.id$ is unnecessary to them.

Full redundancy usually occurs when an IED is configured to publish a data set but the configuration of the intended subscribers is incorrect, or the system design is changed but the configuration does not change accordingly. Partial redundancy happens when the subscribers are only need parts of the data object set. Since a publication may be subscribed by the data consumers which have different demand, it is flexible and convenient to put redundant data objects in one publication. However, because the subscribers can receive the whole payload of the message, it may expose more information to unintended consumers and violate the *principle of least privilege* of information security. It depends on the system's policy to allow the partial redundancy or not in applications.

5.1.3 Source Anomaly

If a subscriber s requests a subscription to a data set ds published by a publisher p, but p does not exist in the system, we say the subscription has a *source anomaly*. Formally,

Definition 5.5. A subscription request $\mathcal{R}_{sub}(s, p, ds)$ has a *source anomaly* if $p \notin \mathsf{E}$.

In the motivating, a subscription of $\mathcal{R}_{sub}(S_1, P_3, \{Op_1, Tr_1\})$ would have a source anomaly since there is no P_3 in the system and the data set $\{Op_1, Tr_1\}$ is actually hosted by P_1.

Source anomalies may occur when the required data sets are moved to other entities and the publisher is removed from the system. But the intended data consumers do not change accordingly.

5.1.4 Data Dissatisfaction

Given a subscription request $\mathcal{R}_{sub}(s, p, ds)$, if no publication (p, ds') can provide all data objects requested in the subscription, we say the subscription is *data dis-satisfactory*. There are two types of data dissatisfaction: *"hard" dissatisfaction* and *"soft" dissatisfaction*.

In the hard data dissatisfaction anomaly, one or more data objects in ds are not published by the publisher p at all. Formally,

Definition 5.6. A subscription request $\mathcal{R}_{sub}(s, p, ds)$ is *hard data dis-satisfactory* if

$$\exists d \in ds : d \notin \overline{\mathcal{R}_{pub}}(p)$$

In the motivating example, a subscription request of $\mathcal{R}_{sub}(S_1, P_1, \{Op_1, Tr_1, P_1.id\})$ would be hard data dis-satisfactory since P_1 does not publish $P_1.id$.

In the soft data dissatisfaction anomaly, one or more data objects in ds are not published in a single publication from p. But the data objects may be contained in other publications from p. If the subscriber requests additional publications, it can get all required data objects.

Definition 5.7. A subscription request $\mathcal{R}_{sub}(s, p, ds)$ is *soft data dis-satisfactory* if

$$[\forall ds' \in \widehat{\mathcal{R}_{pub}}(p)\ ds \nsubseteq ds'] \ \wedge\ [\forall d \in ds\ \exists ds'' \in \widehat{\mathcal{R}_{pub}}(p)\ d \in ds'']$$

For example, a subscription of $\mathcal{R}_{sub}(S_1, P_1, \{Op_1, Tr_1, St_{1,2}\})$ was a soft data dis-satisfactory subscription because no single publication from P_1 is able to satisfy the subscription. But if S_1 subscribed the both publications from P_1, it could get all data it needs. In real applications, however, each publication and each subscription are usually designed for a particular function. It is rare and unreasonable for cross-application subscription. It may violate the principle of least privilege too because the subscriber can get access to unnecessary data objects from multiple subscriptions.

5.2 Anomaly Detection Algorithms

In this section, we present a collection of algorithms that detect the anomalies discussed in Section 5.1. These algorithms are based on the multicast model proposed in Chapter 4, and use the data structures, relations, functions defined in the model.

5.2.1 Detect Ownership Anomaly

As discussed in Section 5.1.1, the ownership anomaly occurs when a publisher publishes a data object which is not owned by it. The algorithm takes a publisher p as an input and returns a data object set namely DSet, which contains the data objects which are not owned by p but published by it. If DSet is empty, the publisher p has no publication ownership anomaly. The pseudo code of the algorithm is shown below as Algorithm 5.1.

The algorithm first creates a list namely DKeys, which consists of the hash values (keys) of each data object d owned by the publisher p (Line 4 through 7). It calculates the hash values of each data object in $\overline{\mathcal{R}_{own}}(p)$ and put them into DKeys. After sorting the list using the quicksort algorithm [16] by hash values (Line 8), it performs the binary search for each data object d' published by p by their hash values (keys). If nothing is searched, the d' is not owned by p and p should have an ownership anomaly. d' will be appended to DSet.

Note when the algorithm is calculating the hash values of data objects, the inputs to the hash function are the identities of data objects, rather than their values. The multicast model, as well as the algorithms discussed in this section, concerns the multicast data flow paths in the configuration files. When implementing the model and the algorithms, we make use of the naming conventions in particular specifications.

Given the size of the data object set of the multicast system is n, the binary search at Line 11 can be

```
Algorithm 5.1: Detect Ownership Anomaly
input  : $p$
output: DSet
begin
    DKeys $\leftarrow nil$;
    DSet $\leftarrow \emptyset$;
    for $d \in \overline{\mathcal{R}_{own}}(p)$ do
        key $\leftarrow$ hash ($d$) ;
        appendKey (DKeys, key) ;
    end
    quickSort (DKeys) ;
    for $d' \in \overline{\mathcal{R}_{pub}}(p)$ do
        key $\leftarrow$ hash ($d'$) ;
        result $\leftarrow$ binarySearch (DKeys, key) ;
        if result = $nil$ then
            appendSet (DSet, $d'$) ;
        end
    end
end
```

determined in $O(\lg n)$ time [16] and the steps from Line 9 through Line 15 can be determined in $O(n{\cdot}\lg n)$ time. That is also the time needed for the whole algorithm.

5.2.2 Detect Publication Redundancy

We design a single algorithm to detect both full redundancy and partial redundancy. The algorithm takes a publication $\mathcal{R}_{pub}(p, ds)$ as an input and needs the support of the consumer set C. It returns a data object set namely RDSet, which is used to store the redundant data objects in ds. The algorithm also returns the status of the publication: full-redundancy, partial-redundancy or clear. The pseudo code of the algorithm is shown as Algorithm 5.2.

The usage of DKeys is same as Algorithm 5.1. The algorithm first calculates the hash values of all data objects that are consumed in the system, and then stores the hash values (keys) in DKeys (Line 5 through 10). To improve the algorithm efficiency, DKeys is also sorted using the quicksort algorithm (Line 11). From Line 12 to Line 18, the algorithm calculates the hash values of each data object d published in $\mathcal{R}_{pub}(p, ds)$ and searches the data object in DKeys using the binary search algorithm. If the search fails, the data object d should be a redundant data object and is inserted to the redundant data object set RDset. Line 4 and Line 19 count the number of the data objects in the published data objects set ds and the redundant data object set RDset respectively. If the two numbers rsetn and psetn equal to each other, all data objects in ds are redundant and it is a full redundancy anomaly. If RDset is empty and rsetn is 0, nothing is redundant and the publication status is clear. Otherwise it is partial-redundancy (Line 20 through 26). The algorithm is determined in $O(n^2)$ time because the steps from Line 5 to Line 10, which initialize DKeys, are determined in $O(n^2)$ time.

Algorithm 5.2: Detect Publication Redundancy

```
   input  : $\mathcal{R}_{pub}(p, ds)$, C
   output: RDSet, status
1  begin
2      DKeys ← $nil$;
3      RDSet ← $\emptyset$;
4      psetn ← countSet ($ds$) ;
5      for $c \in$ C do
6          for $d \in \overline{\mathcal{R}_{con}}(c)$ do
7              key ← hash ($d$) ;
8              appendKey (DKeys, key) ;
9          end
10     end
11     quickSort (DKeys) ;
12     for $d \in ds$ do
13         key ← hash ($d$) ;
14         result ← binarySearch (DKeys, key) ;
15         if result = $nil$ then
16             appendSet (RDSet, $d$) ;
17         end
18     end
19     rsetn ← countSet (RDSet) ;
20     if rsetn = psetn then
21         status ← full-redundancy ;
22     else if  rsetn = $0$ then
23         status ← clear ;
24     else
25         status ← partial-redundancy ;
26     end
27 end
```

5.2.3 Detect Source Anomaly

This algorithm takes a subscription request $\mathcal{R}_{sub}(s, p, ds)$ and the whole entity set E as the inputs, and returns the checking result of the subscription: source-anomaly or clear. The pseudo code is shown as

Algorithm 5.3.

The algorithm calculates the hash values of all entities in the system, and then stores the hash values (keys) in a list named EKeys (Line 3 through 6). After sorting the list by keys using the quicksort algorithm (Line 7), the algorithm searches the list for the key of the publisher p (Line 8 and Line 9). If the search fails, this subscription has a source anomaly, otherwise its status is cleared (Line 10 through 14). The algorithm is determined in $O(n)$ time. The list EKeys can be used by other algorithms or other applications.

Algorithm 5.3: Detect Source Anomaly

```
   input  : $\mathcal{R}_{sub}(s, p, ds)$, E
   output: status
1  begin
2      EKeys ← $nil$;
3      for $e \in$ E do
4          key ← hash ($e$) ;
5          appendKey (EKeys, key) ;
6      end
7      quickSort (EKeys) ;
8      key ← hash ($p$) ;
9      result ← binarySearch (EKeys, key) ;
10     if result = $nil$ then
11         status ← source-anomaly ;
12     else
13         status ← clear ;
14     end
15 end
```

5.2.4 Detect Data Dissatisfaction

We design a single algorithm to detect both hard-dissatisfaction and soft-dissatisfaction anomalies. It takes a subscription $\mathcal{R}_{sub}(s, p, ds)$ as input and returns the checking result as: hard-dissatisfaction, soft-dissatisfaction or clear.

To improve the efficiency, we create two macros `quickSortSet` and `binarySearchSet`. The macro `quickSortSet` is designed for sorting a set by the hash values of members using the quicksort algorithm. The members of the set could be data objects or entities. The hash function takes members' identities, rather than values (if members are data objects) as inputs. The macro `binarySearchSet` is designed for searching a set for a particular member by the hash values of set members and the target using the binary search algorithm. Usually, the set is already sorted by hash values of members' identities.

Actually the steps of the two macros are already presented in previous algorithms. The variable PubSet

Algorithm 5.4: Detect Data Dissatisfaction

```
input  : $\mathcal{R}_{sub}(s, p, ds)$
output: status
begin
    PubSet ← $\emptyset$;
    pubsetn ← $0$ ;
    subsetn ← countSet ($ds$) ;
    satisfied ← true ;
    for $ds_p \in \widehat{\mathcal{R}_{pub}}(p)$ do
        quickSortSet ($ds_p$) ;
        for $d \in ds$ do
            result ← binarySearchSet ($ds_p$, $d$) ;
            if result = $nil$ then
                satisfied ← false ;
            else
                appendSet (PubSet, $d$) ;
            end
        end
        if satisfied = true then
            status ← clear ;
            break ;
        else
            satisfied ← true ;
        end
    end
    pubsetn ← countSet (PubSet) ;
    if status $\neq$ clear then
        if pubsetn $<$ subsetn then
            status ← hard-dissatisfaction ;
        else if pubsetn = subsetn then
            status ← soft-dissatisfaction ;
        end
    end
end
```

stores the data objects which s subscribes in this subscription, and p does publish. The size of PubSet is represented by pubsetn. The variable subsetn is the number of the data objects required in this subscription, *i.e.* the size of ds. By comparing pubsetn and subsetn, we can know if all required data objects in ds are provided by p. The variable satisfied is a flag indicating if the subscription request $\mathcal{R}_{sub}(s, p, ds)$ can be satisfied by a single publication from p, *i.e.* if there exists a data object set, which is the superset of ds and published by p. The default value of satisfied is **true**.

The core of the algorithm is from Line 6 to Line 22. It checks all data object sets published by p to see if there is a publication can provide all data objects the subscription requires, or if all data objects in ds are provided by p's publications. For each data object set ds_p from p, the algorithm first sorts it using `quickSortSet` to improve search efficiency (Line 7). Then it searches the set for each data object in ds using `binarySearchSet` (Line 9). If the search fails (Line 10), it means ds is not the subset of the current ds_p and the publication cannot satisfy the subscription (Line 11). Otherwise, the searched data object d is appended to PubSet. The loop will not break even if a search fails. It needs to guarantee that every data object in p's publications is checked. If the variable satisfied still keeps **`true`** after the inside loop finishes, the subscription $\mathcal{R}_{sub}(s, p, ds)$ can be satisfied by current publication $\mathcal{R}_{pub}(p, ds_p)$. Its status will be set as **`clear`** and the outside loop can break (Line 16 through 21). Otherwise, the variable satisfied will be reset and the outside loop continues until all data sets published by p (ds_p) are searched.

After the search finishes, we get the final PubSet, which contains all data objects p publishes. The size of PubSet is obtained at Line 23. Note that the duplicated data objects are already removed. If the status is not **`clear`**, the algorithm checks pubsetn (Line 23 through 30). If pubsetn is less than subsetn, some data objects in ds are not included in p's publications. The subscription has a **`hard-dissatisfaction`** anomaly. If the two numbers equal, this is a **`soft-dissatisfaction`** anomaly.

Given the size of the data object set of the multicast system is n and Assumption 4.3, there are at most n publications from p. So the algorithm is determined in $O(n^2 \cdot \lg(n))$ for the worst case.

5.3 Discussion

As mentioned in Chapter 1, the data are the focus of a publication-subscription system. They connect all group members in a multicast application. The four anomaly detection algorithms track the path of a data flow, from the data owners to the data consumers. The errors or missing components on the flow will be detected. However, if the whole data flow is incorrect or there are more than two anomalies in a publish-subscribe data flow, the algorithms may not be able to detect them. There are two basic scenarios:

Redundant multicast group A multicast group is not needed anymore due to the updated application logic, but the engineers do not remove the relevant publication, subscriptions and data objects from the configuration file. Because the redundant data flow is still complete, the algorithms cannot detect the anomaly.

This usually occurs when the system functions are re-allocated to IEDs and some old functions need to be removed from certain devices.

In the motivating example, if the substation does not require P_2 to publish $\{St_{2,1}, St_{2,2}, St_{2,3}\}$, and S_2, S_4 do not need to monitor the status data either, then the publication $\mathcal{R}_{pub}(P_2, \{St_{2,1}, St_{2,2}, St_{2,3}\})$, the subscriptions $\mathcal{R}_{sub}(S_2, P_2, \{St_{2,1}, St_{2,2}, St_{2,3}\})$ and $\mathcal{R}_{sub}(S_4, P_2, \{St_{2,1}, St_{2,2}, St_{2,3}\})$ should be removed. However, if these relations are still kept in the configuration, no error message will be reported.

Collusive anomalies The algorithms in Section 5.2 aim to detect isolated anomalies.. If a publication and its correlated subscriptions have more than one anomaly, these anomalies may complement each other and hide the mistakes. Under this circumstance, the whole configuration looks correct and the algorithms are not able to detect the anomalies. We call these anomalies *collusive anomalies*.

In the motivating example, according to the application logic, P_1 should not publish $P_1.id$ in $\mathcal{R}_{pub}(P_1, \{Op_1, Tr_1\})$, or a *publication redundancy anomaly* will be detected. However, if a *data dis-satisfactory* subscription request $\mathcal{R}_{sub}(S_1, P_1, \{Op_1, Tr_1, P_1.id\})$ is inserted to the configuration file, the both abnormal publication and subscription will not be detected although either of them can be detected alone.

In the other word, this work can derive group memberships from application logic by configuration files, and detect multicast configuration anomalies by analyzing publication-subscription data flow. It cannot detect the anomalies which are not consistent with the application logic but do not break a data flow.

Furthermore, our attention is focused on application layer anomalies, the mistakes in the network layer configuration, like duplicated multicast addresses for different multicast groups, are out of the scope of this work. We also do not distinguish underlying reasons of the anomaly. How to eliminate the anomalies is also out of the scope of this work.

Chapter 6

Implementation and Case Study

In this chapter, we present the system implementation of the application-aware, derived group approach for power grid multicast communications. The basic idea is to derive group memberships and publication-subscription relationships based on data dependencies, which are extracted from an appropriate extension of system domain-specific specifications. We take advantage of the multicast model in Chapter 4 and the configuration verification algorithms in Chapter 5 to analyze and verify the validity of multicast groups and publication-subscription configurations. A multicast and group key management architecture based on the Group Domain of Interpretation (GDOI) [10] is designed and then used to set up group security associations based on the data dependencies and consistency analysis results. We show that the challenges of manageable configuration discussed in Chapter 1 can be overcome by linking network layer secure multicast configuration to application-specific configuration of power substations.

To demonstrate this methodology we have developed a prototype system *SecureSCL* with a case study of secure GOOSE in IEC 61850 power substation networks. SecureSCL extends SCL by integrating new elements representing IPsec multicast, security credentials and group key servers[1]. The multicast groups of GOOSE are extracted from the extended SCL specifications. SecureSCL transforms derived group information and security extensions to the multicast model presented in Chapter 4 and checks the configuration consistency and correctness. The prototype system is validated by using it on a portion of the SCL specification of an experimental substation of the Tennessee Valley Authority (TVA).

6.1 Basics

Before introducing the details of our approach and the system design, we first clarify the application assumptions and the design principles.

[1]Since the design and implementation in this chapter is heavily based on IEC 61850 and SCL, please refer to Chapter 2 for background details.

6.1.1 Assumptions

Based on the observation of multicast applications in power grid systems, especially power substation networks, we consider a class of applications with the following characteristics:

- Multicast packets are transmitted by UDP/IP. Usually multicast is supported in the link layer and the network layer using dedicated multicast addresses. Our attention in this work is focused on network layer multicast, *i.e.* IP packets with Class-D addresses. Beyond the IP protocol, UDP is one of the most widely used transport protocols for multicast applications. Therefore, we base the design in this chapter and the experiments in Chapter 7 on UDP/IP protocols.

- In a multicast group, the data flow from the sender to the recipients is one-way, *i.e.* there is no feedback mechanism required. In each communication session, the application logic does not require recipients to reply the sender's message with meaningful responses. No hand-shake process is needed[2]. This is true for many multimedia stream applications, as well as power grid communication applications like GOOSE and SMV.

- Within a particular multicast group there is only one publisher or sender, *i.e.* only a single data source. The publisher launches a multicast group and sends messages to the group. The interested entities will join the group and listen to the publisher only. This also implies that the roles of the publisher and subscribers do not change in a group. Each group only represents one single application. If the publisher is interested in the data from other entities, it can join the group as a role of a subscriber. But its role in a single group does not change. This is true for multicast applications in power substation networks.

- Multiple multicast groups may exist in a network simultaneously. As mentioned above, each data source or publisher can launch an individual group and these groups will be set up for different applications at the same time. For example, there are more than 40 groups in TVA Bradley IEC 61850 substation.

[2]In the experiments of Chapter 7, the recipients respond the sender's request with an acknowledgement message. But this is just used to measure round trip latencies of IPsec multicast. They are not required by application logic and have no semantic meanings.

6.1.2 Design Principles

We propose four key design principles that structure possible architectures. They are: defense-in-depth, low latency, standardized protocols, derivability. We discuss each in turn.

Defense in depth Many existing power grid networks rely exclusively on firewalls, gateways, or virtual LANs to isolate largely unprotected control elements from public networks or enterprise networks. However, the perimeters are not always trustworthy. A 'good' control device may be compromised due to a fault [15], operators sometimes attach compromised machines to the LAN, and wireless systems may admit on or near-site intruders if compromised or misconfigured. Misconfigured firewall rules could cause firewalls/gateways to fail or bypassed and VLANs are not designed for security protection. Better defense in depth is obtained if control devices have built-in secure communication modules and do not automatically trust other elements that manage to communicate within the perimeter.

Low latency As mentioned in various process control applications have real-time challenges to existing security protocols. Timing constraints must be studied when designing a security system for process control networks and the performance must be tested or verified before deployment. Basic methods include avoiding public-key cryptography signature and verification for every packet, and running key exchange protocols before transmission starts.

Standardized protocols A number of provably-secured protocols have been designed, inspected and implemented by network security communities. It is risky and probably unnecessary to design a new protocol from scratch. Standardized protocols and technologies also save time and cost for system deployment since they are more likely to have existing implementations.

Derivability Industry has developed a number of configuration tools for particular applications like SCL. Integration with existing configuration tools enables security tools to automatically derive information about the system being secured, such as network topology and details of application logic. This enables richer security features, such as individual source authentication. Furthermore, it provides the possibility of using existing configuration analysis tools for verifying the correctness of the configuration or detecting errors. The integration would also ease the deployment of security solutions by avoiding exclusive security configuration.

6.2 Overview

6.2.1 Network Model

We present the network model, which corresponds to the multicast model presented in Chapter 4. The multicast model emphasizes the data dependencies between multicast group members at the application layer, while the network model is focused on the group topology and data flow in the network layer.

Figure 6.1 illustrates a basic power substation multicast group. In such a group, there is one group

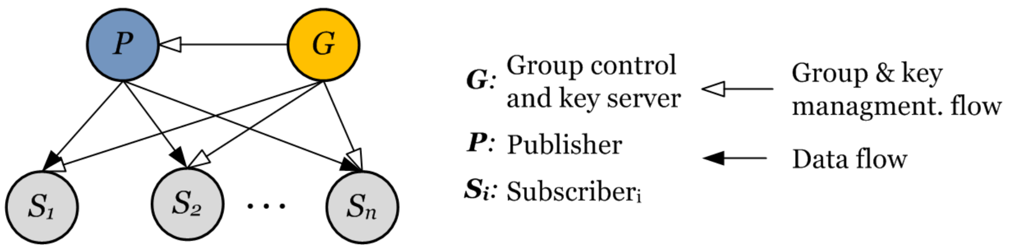


Figure 6.1: Network Model

controller & key server G, one publisher P, and a number of subscribers S_i. The group controller & key server, or group controller in short, G enforces group authorization policy and distributes group keys to P and S_i by group key exchange protocols. Multicast messages are transmitted from P to S_i through the substation LAN, which is usually a switched Ethernet. According to the assumptions in Section 6.1.1, the application logic within a multicast group does not change, and the roles of P and S_i keep constant.

There are two types of data flow in the group. The data flow is used to transmit application data from the publisher to the subscribers. The group & key management flow is used to distribute group keys or other security credentials from the group controller to the group members. These two types of flow can be implemented using the same protocols or share parts of a protocol suite.

We assume G is secure and trusted. Any credentials, key materials and policy decisions made by G are credible. We also assume configuration files distributed among G, P and S_i are authentic. The mechanisms for securely distributing configuration files are orthogonal to this work. The architecture does not interfere with the routing mechanism of data packets. For sake of simplicity of design, we assume the multicast traffic only occurs within a control system LAN and the key management protocol has reliable communication.

6.2.2 Architecture

Figure 6.2 shows the host architecture of a multicast group member P or S_i. As in many network systems

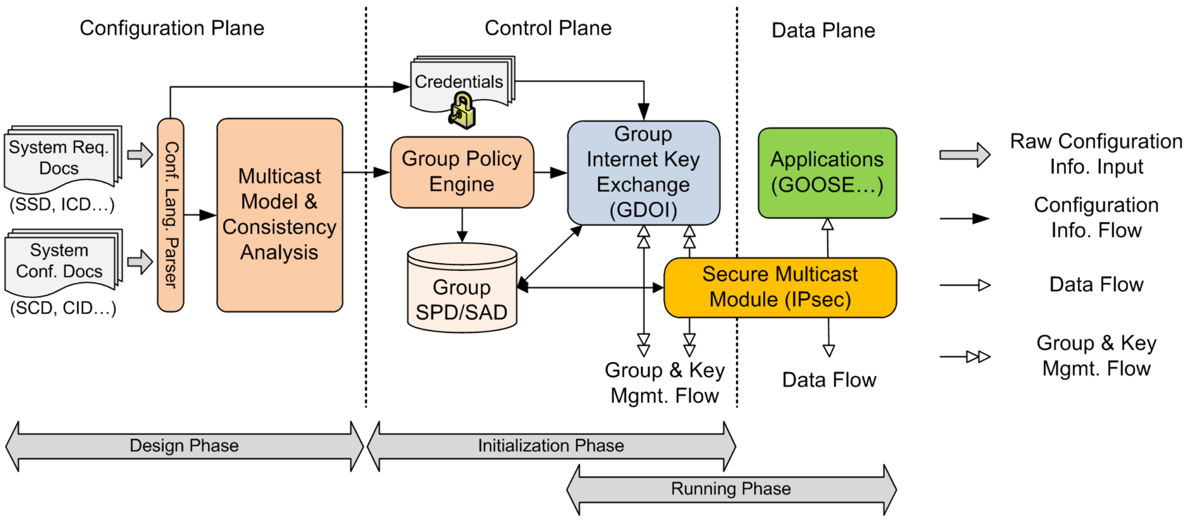


Figure 6.2: System Architecture

like [13], the system is partitioned into three planes: *configuration plane*, *control plane* and *data plane*. Furthermore, the system works in three phases: *design phase*, *initialization phase* and *running phase*.

The configuration plane works in the design phase. A configuration language parser parses system specification and configuration files, like SCD files. Based on the parser's output, a multicast model and consistency analyzer sets up the data model and the publish-subscribe model presented in Chapter 4. It also checks configuration correctness and consistency using the algorithms discussed in Chapter 5. If a configuration anomaly is detected, the system will go back to the step of configuration revision. If the verification succeeds, the system enters the initialization phase and the control plane takes over. An illustrative diagram about the working phases of the system is shown in Figure 6.3.

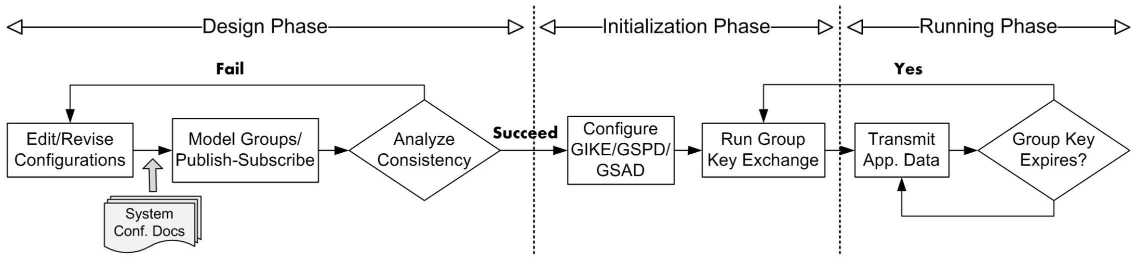


Figure 6.3: System Working Phases

The control plane is in charge of group and key management. A group policy engine (GPE) extracts group association information and security configuration from the multicast model and the original security extended configuration files (see Section 6.3). It retrieves pre-installed credentials like pre-shared keys or certificates, configures the Group Internet Key Exchange (GIKE) module, and then triggers group key exchange between the group controller and the group members. The traffic used to exchange group keys is called *group key & management flow* (see the network model in Section 6.2.1). Credentials like session keys and relevant negotiated policies for incoming and outgoing multicast traffic are inserted and stored in Group Security Policy Database (GSPD) and Group Security Association Database (GSAD). After the group key exchange finishes, the system enters the running phase and the data plane starts working. Note that the control plane continues working during the running phase for refreshing shared group keys.

The data plane functions are straightforward. It is comprised of upper layer applications, like GOOSE, and Secure Multicast Module (SMM). Incoming and outgoing application packets will be processed by the SMM (in our implementation this is IPsec) according to the GSPD and GSAD. The traffic used to transmit securely protected application packets is called *data flow* (see the network model of Section 6.2.1). At the same time, the GIKE module also makes use of the SMM module to securely refresh group session keys periodically without interfering the data flow. So both the control plane and the data plane works during the running phase,

The host architecture of G is almost same as regular group members' architecture shown in Figure 6.2. The major difference is the GPE module. For a regular group member, GPE only tells GIKE the information about the multicast group the host needs to join, the group controller and key server and the basic configuration profiles for running the group key exchange protocol like the references to security credentials. For G, the GPE further directs GIKE for group authorization. Based on the group associations derived from the multicast model, GPE provides GIKE with the group information like the multicast groups addresses (if there are multiple groups in one network), the identities of the group members that are allowed to join the group and their security credentials like public key certificates, and parameters for group management like the interval of refreshing group keys, *etc*. Unlike P and S_i, the SMM module in G is not used for protecting the data flow. Instead, it is just used to protect the key management flow.

The whole system is implemented using C/C++ on Fefora 7. The extended configuration language parser is developed using libxml [74]. The Group Policy Engine module makes use of NETLINK sockets to manipulate IPsec SPD and SAD. A reference implementation of GDOI from Cisco is used for the Group

Internet Key Exchange module.

In the next sections, we will give a detailed introduction of each component in the architecture.

6.3 Extended Configuration Language

A full-fledged control system configuration language like Substation Configuration Language (SCL) provides a global view of the whole control network. It not only defines data structures, functionalities and default values of control devices' parameters but also specifies network topology and communication associations between devices. By analyzing a configuration file, we can obtain all information about multicast applications within a power substation network, including network information for multicast, publisher-subscriber associations and multicast message sources and payload data structures.

To support secure multicast, especially IPsec-based multicast, the configuration language needs to be extended for more information, including: 1) credentials (or their references) required for group key exchange; 2) additional networking entities which facilitate secure communications, like a group key server.

As an application-specific process, configuration extension heavily depends on the configuration mechanisms and the configuration file format. For those structured and standardized configuration languages, such as SCL, a number of XML-based security specifications and configuration tools can be integrated for IPsec policy specification [47, 76] and credential description[8].

In this work, we base the implementation on SCL. We show the extension to the network configuration to raise GOOSE messages to the network layer, the integration of security information, especially the credentials used for group key exchange, and the specification of the group control and key server.

6.3.1 Communication Interface

As introduced in Section 2.3.1, SCL provides the element Communication and its child elements like ConnectedAP (connected access point) for describing all information about the network connections between IEDs. It includes the parameters of a control device's network interfaces, like the IP address, the subnet address and the gateway address, and multicast network parameters like the link layer multicast addresses (MAC) for GOOSE applications *etc*. To support the network layer GOOSE messages, we extend the ConnectedAP element and enable the IP multicast configuration.

Figure 6.4 shows the parts of the extended ConnectedAP element. It specifies the network interface of a protective relay named IED1. The IP address and the IP network information are specified in Line 4 through

```
 1  <Communication>
 2    ...
 3    <ConnectedAP apName="apIED1" desc="Trip Publisher AP" iedName="IED1">
 4      <Address>
 5        <P type="IP">192.168.1.20</P>
 6        <P type="IP-SUBNET">255.255.255.0</P>
 7        <P type="IP-GATEWAY">192.168.1.1</P>
 8        ...
 9      </Address>
10      <GSE cbName="gcbTrip" ldInst="PROT">
11        <Address>
12          <P type="IP">224.0.0.4</P>
13          ...
14          <!--<P type="MAC-Address">01-0C-CD-01-01-46</P>-->
15        </Address>
16      </GSE>
17    </ConnectedAP>
18    ...
19  <Communication>
```

Figure 6.4: Extended Multicast Network Configuration

9. The original purpose of the Address element is to describe the IED's ACSI interface using OSI protocols stack. In SecureSCL, it is also used for IPsec multicast. The GIKE module uses this address for running group key exchange protocol. If the IED works as a publisher in the whole network and launches a multicast group, the outgoing multicast packets will be encapsulated using IPsec with the source address specified by this element (192.168.1.20 in this example.)

The element GSE is designed for publishing GOOSE messages. Each GSE is used for one GOOSE application and an IED may have multiple GSE elements, *i.e.* an IED is able to publish more than one type of GOOSE messages. GSE is associated with one logical device's GSE control block. In this example, it describes the multicast network interface for the logical device PROT's control block cbName. Usually, a MAC address is used for a GOOSE multicast group (Line 14). To raise GOOSE to the network layer, we revise the original design and replace the MAC address with a Class-D IP address (Line 12). This will be the IP multicast group address for all subscribers. The group controller G will also use this address for group membership management.

6.3.2 Security Extension for Credentials

IEC 61850 is focused on power functionalities and utility communications. It neither addresses the security of network communication nor provides sophisticated mechanisms for specifying security information. To enable secure multicast, especially IPsec based multicast, we extend SCL by integrating security credentials description into AccessPoint elements.

We make use of the element KeyInfo , which is defined in XML Signature [8], to describe security credentials. The element KeyInfo is an optional element in the XML Signature specification. It enables the recipient(s) to obtain the keys needed to validate the signature. KeyInfo may contain keys, names, certificates and other public key management information. Since the XML Signature is one of the most widely used standards for XML cipher processing, KeyInfo also becomes the de facto standard for cryptographic key description in XML.

Figure 6.5 shows the security extension to the element IED in SecureSCL. An access point is a logical communication interface of an IED's logical devices to a substation network. The information about its

```
 1  <IED name="IED1" type="SecureIED" desc="Protective Relay">
 2    <AccessPoint name="apIED1" desc="Trip Publisher AP">
 3      <Server>
 4        <Authentication certificate=”true” none="false" strong="true"/>
 5        ...
 6        <GSEControl appID="TripGoose" datSet="dsTripLogic" name="gcbTrip"/>
 7      </Server>
 8      <ds:KeyInfo Id="IED1cert">
 9        <ds:X509Data>
10          <ds:X509Certificate>HX...</ds:X509Certificate>
11        </ds:X509Data>
12      </ds:Keyinfo>
13    </AccessPoint>
14  </IED>
```

Figure 6.5: The Extension of Security Credentials

underlying physical network ports and network protocols is defined in the element of ConnectedAP (see Figure 6.4). We insert the element KeyInfo as a child element of AccessPoint (Line 8 through 12) so that it can serve all communication protocols via this access point. In this example, an X509 certificate is embedded into the configuration file. It can be used for group key negotiation in multicast applications. SCL defines a child element Authentication (Line 4) for the element Server. It is used to indicate the authentication mechanisms used for the access to the data services with the Server. However, SCL does not provide any explanation about how to use it or implement it. In SecureSCL, we take advantage of the element to indicate that certificates are used for authentication.

6.3.3 Group Controller & Key Server

To support group key management and implement the architecture proposed in Section 6.2.2, we create an element named GCKS in the extended SCL to describe the group controller and key server.

Figure 6.6 shows the parts of an extended SCL file with the extension of the element GCKS (Line 6

```
 1 <SCL xmlns="http://www.iec.ch/61850/2003/SCL
 2      xmlns:sscl="http://seclab.illinois.edu/SecureSCL"
 3      xmlns:ds="http://www.w3.org/2000/09/xmldsig# ...>
 4   <Communication>
 5     <SubNetwork name="TVASecSubnet" type="SecSubnet">
 6       <sscl:GCKS desc="Group Controller and Key Server" Id="GCKS1">
 7         <Address>
 8           <P type="IP">192.168.1.2</P>
 9         </Address>
10         <sscl:GIKE>
11           <sscl:GroupProtocol>GDOI</sscl:GroupProtocol>
12           <sscl:Port>848</sscl:Port>
13         </sscl:GIKE>
14         <ds:KeyInfo Id="GCKS">
15           <ds:X509Data>
16             <ds:X509Certificate>MI...</ds:X509Certificate>
17           </ds:X509Data>
18         </ds:Keyinfo>
19       </GCKS>
20       ...
21     </SubNetwork>
22   </Communication>
23   ...
24 </SCL>
```

Figure 6.6: The Element of GCKS

through 19). Because a group key server usually serves the whole substation network, we set it as a child element of SubNetwork. The element is comprised of three parts. First of all, an IP address is assigned to the group controller (Line 7 through 9). All group members will use this address for running the group key exchange protocol with the group controller. Secondly, a particular group key protocol is specified in Line 10 through 13. The architecture in Section 6.2.2 does not restrict the type of group key protocols and multiple options are allowed. In our work, we choose the GDOI with its default port number 848. Finally, the group controller's X509 certificate is embedded from Line 14 through 18.

The extended SCL files provide sufficient support of secure multicast configuration at the network layer. In the next section, we will focus on the application layer. We will show how to derive multicast groups by setting up publish-subscribe relationship and identifying valid publishers and subscribers from functional configurations.

6.4 Multicast Modeling Based on SCL

In this section, we show how to map the SCL data object model to the multicast model and derive multicast groups. We don't change or extend the SCL specification for this step although more efficient and error-resistant mechanisms could be introduced to SCL. Based on the derived multicast model, we implement the anomaly detection algorithms discussed in Chapter 5. If the configuration verification fails, the configuration should be reviewed and corrected. Otherwise the initialized model will be used by the Group Policy Engine to configure the group key exchange protocol.

6.4.1 Ownership

An IED can be considered as a combination of logical nodes and their data objects. According to Definition 4.1, all logical nodes and their data objects should be owned by the hosting IED.

Figure 6.7 shows the data structure of a protective relay. The element LDevice defines the visible and

```
 1 <IED name="IED1" type="SecureIED" desc="Protective Relay">
 2   <AccessPoint name="apIED1" desc="Trip Publisher AP">
 3     <Server>
 4       ...
 5       <LDevice inst="PROT">
 6         ...
 7         <LN inst="1" lnClass="PDIS" lnType="IED1-PDIS-Type"/>
 8         <LN inst="1" lnClass="PTRC" lnType="IED1-PTRC-Type"/>
 9       </LDevice>
10     </Server>
11   </AccessPoint>
12 </IED>
13 ...
14 <DataTypeTemplates>
15 ...
16   <LNodeType id="IED1_PTRC_Type" lnClass="PTRC">
17     ...
18     <DO name="Tr" type="tPTRC_TrOp"/>
19     <DO name="Op" type="tPTRC_TrOp"/>
20     ...
21   </LNodeType>
22 </DataTypeTemplates>
```

Figure 6.7: Ownership in SCL

accessible logical nodes within an IED. In this example, the IED "owns" two logical nodes PDIS1 and PTRC1 (Line 5 through 9). The data objects of these two logical nodes can be identified by checking their classes. Besides, the logical nodes can be can customized in the element DataTypeTemplates (Line 16 through 21). By checking these definitions we can derive the ownership relation between IEDs and data objects.

6.4.2 Publication

Figure 6.8 shows an example of GOOSE configuration in SCL. Each LDevice element within an IED has

```
 1 <IED name="IED1" type="SecureIED" desc="Protective Relay">
 2   <AccessPoint name="apIED1" desc="Trip Publisher AP">
 3     <Server>
 4       ...
 5       <LDevice inst="PROT">
 6         <LN0 lnClass="LLN0" lnType="IED1-LLN0-Type">
 7           <DataSet name="dsTripLogic">
 8             <FCDA daName="general" doName="Tr" ... ldInst="PROT" lnInst="1"/>
 9             <FCDA daName="t" doName="Tr" ... ldInst="PROT" lnInst="1"/>
10             <FCDA daName="general" doName="Op" ... ldInst="PROT" lnInst="1"/>
11             <FCDA daName="t" doName="Op" ... ldInst="PROT" lnInst="1"/>
12             ...
13           </DataSet>
14           <GSEControl appID="TripGoose" datSet="dsTripLogic" name="gcbTrip".../>
15         </LN0>
16       ...
17       </LDevice>
18     </Server>
19   </AccessPoint>
20 </IED>
```

Figure 6.8: Publication in SCL

a LLN0 logical node (Line 6), which represents common data and features of a logical device. It usually contains a number of DataSet elements. The DataSet represents a collection of data attributes from multiple data objects, which could be the message payloads of a GOOSE message (Line 7 through 13). Each FCDA element within the DataSet specifies the detailed information about these data attributes.

LLN0 also contains an element of GSEControl, which specifies the parameters of a GOOSE application using the element's attributes. One important attribute is datSet, which indicates which data set is published by the GOOSE message. In this example, the data set dsTripLogic should be published as a payload of the GOOSE message. Using the name attribute of the IED, AccessPoint and GSEControl, we can associate the application layer configuration with the network layer parameters in the Communication and corresponding ConnectedAP elements.

In summary, based on above configuration information, we can derive: IED1 publishes the data object set dsTripLogic using network layer GOOSE messages. The source address of each packet is 192.168.1.20 and the multicast destination address is 224.0.0.4.

6.4.3 Consumption & Subscription

According to the assumptions in Section 4.3.1, if an IED requires a number of data objects which are published by a GOOSE message, the IED has to subscribe to the publication. Such feature is realized by the elements Inputs and ExtRef.

Figure 6.9 shows parts of SCL configuration of IED2, a switchgear which hosts a circuit breaker (XCBR,

```
 1  <IED name="IED2" desc="Switchgear" type="SecureIED">
 2      ...
 3    <AccessPoint name="apIED2" desc="IED2 GOOSE Trip Subsriber AP">
 4      <Server>
 5        <LDevice inst="CTRL">
 6          <LN desc="CircuitBreaker" inst="1" lnClass="XCBR" lnType="IED2-CTRL-XCBR">
 7            <Inputs>
 8              <ExtRef daName="general" doName="Tr" iedName="IED1" ldInst="PROT" .../>
 9              <ExtRef daName="t" doName="Tr" iedName="IED1" ldInst="PROT" .../>
10              <ExtRef daName="general" doName="Op" iedName="IED1" ldInst="PROT" .../>
11              <ExtRef daName="t" doName="Op" iedName="IED1" ldInst="PROT" .../>
12            </Inputs>
13          </LN>
14        <\LDevice>
15      <\Server>
16    <\AccessPoint>
17  <\IED>
```

Figure 6.9: Subscription in SCL

Line 6). The logical node XCBR on IED2 requires four data attributes (data objects in the multicast model) from IED1. This consumption/subscription request is specified by the elements Inputs and ExtRef from Line 7 through 14.

By searching the configuration file for the required data objects and attributes in the ExtRef elements, the subscriber should be able to locate the IED which publishes the data. By the relevant elements in Communication and ConnectedAP, the subscriber can figure out the multicast group it should join.

In summary, we can set up the publish-subscribe model from original SCL files, and then run the anomaly detection algorithms to correct configuration mistakes or even improve the functional design.

On the other side, after loading the configuration file, an IED can get all necessary information about the multicast group, including the group controller, the protocol used for group key exchange and relevant security credentials. It is also assigned an IP address for the group key exchange. If the IED is publisher, it is also assigned a multicast address for launching a multicast group. If it is a subscriber, it can join the group immediately. The group controller can get the information like the number of multicast groups, the valid members of each group, and the multicast address for each group. By running group key exchange protocol with each member, it can authorize entities the access privileges to particular groups, or reject joining requests by terminating group key exchange sessions.

Thus the system achieves the automatic multicast group configuration at the network layer by the information from the application layer and mitigates the risk of inconsistent configuration due to human errors.

It significantly improves the system efficiency.

6.5 Group Key Management

6.5.1 Group Policy Engine

The GPE transforms the multicast model to group authorization policy and traffic policy. Authorization policy specifies which entity or IED can join the group and share group keys. It is used by group controllers for group membership management. Traffic policy is used to enforce security services, such as signing and verifying signatures, on individual packets. It is usually set up after the GIKE module finishes a group key exchange. Traffic policy is queried by the SMM module when it is processing multicast packets.

The functionalities of the GPE modules on G and P(or S_i) are different. On G, the GPE module works as a group authorization center. Given the multicast model is already verified during the design phase, the GPE transforms the model to a configuration file recognizable to the GIKE (GDOI), and invokes the GIKE module listening to group key negotiation requests, *i.e.* group join request. After loading the file, the GIKE module has a big picture of all multicast groups. If an IED sends a join request to a wrong group, the GIKE will reject the request by terminating the group key exchange protocol.

The functionalities of the GPE modules on group members are comparatively simple since it is unnecessary for a group member to know all groups. The GPE on group members invokes the host starting group key negotiation with the group controller. Based on the configuration information from application logic, the GPE and GIKE module on P generates traffic policies for outgoing packets, while the modules on S_i consider traffic policies for incoming packets.

6.5.2 Group Internet Key Exchange

The GIKE module is a protocol used for group membership authorization and group key management. In this paper, we have borrowed the idea of a multicast group key management architecture from [29] and [9], and take advantage of the GDOI [10], a centralized multicast security and key management protocol, to perform the task. As a mature protocol, the GDOI is integrated with IPsec protocol suite smoothly, which makes the system design and implementation easy and efficient. Because the network topology of a substation network is relatively stable and the group members rarely join or leave the group when the system is running, we argue that the GDOI is competent to handle group key management in this case.

We now outline the group and key management flow in more details.

- *Initialization*: GPE on G sets up authorization policies, configures and invokes GIKE to load required credentials from extended SCL configuration files and listen to *Join* requests.

- *Join*: GPE on P or S_i invokes GIKE to join a multicast group with the group information and the required credentials. GIKE starts the group key exchange protocol with G and sets up Registration SA. Upon receiving and authenticating a request, GIKE on G queries GSPD to check if it is an authorized member or not. If it is, Data SA and Rekey SA are set up and session keys are distributed. Otherwise the request will be rejected and the key exchange will halt.

- *Key update*: Key update messages are generated and operated by GIKE automatically. The update interval should be specified in the security extended configuration files and configured by GPE. There are two ways to refresh session keys: unicast and multicast. By the unicast way, G has different Rekey SAs with each group member and pushes refreshing keys to group members individually. The unicast key update is convenient for members leaving and revoking a member just by removing corresponding Data SAs and Rekey SAs, and update GSPD. However, it is hard to keep GSAs synchronized especially in a large group. The multicast key update is efficient for GSA synchronization. But it is challenging to guarantee that revoked members cannot access the group any longer.

- *Leave* and *Revoke*. Usually, a group communication system needs to handle the issue of member leave and revocation. In control networks like power substation networks, however, the number of control devices and the network topology is almost fixed for a long period time. Once such a network is initialized, it rarely changes. Therefore, the event of *Leave* and *Revoke* almost never occurs. SecureSCL is based on static group configurations. The group authorization is determined during the design phase. Once the system enters the running phase, we assume the group members, *i.e.* IEDs, will work stably for long time. Therefore, although the GDOI provides methods like de-registration for dynamic group member management, SecureSCL does not make use of it.

6.6 Secure Multicast Module

6.6.1 IPsec-based Multicast

We have based our design on the IPsec protocol suite. IPsec implementations on most off-the-shelf operating systems are able to protect multicast packets natively [5, 11]. If the destination IP of an IPsec packet is a multicast address, hosts joining the multicast group with appropriate SAs and SPs are able to deliver the packet. Such mechanism avoids the packet replication that occurs in the hub-and-spokes schemes like [12] and [69], and guarantees all recipients can receive the message simultaneously(see Chapter 7 for details).

We have discussed the limitations of some link layer security solutions, like IEC 62351 and IEEE 802.1AE in Section 3.1. In this section, we focus on the advantages of IPsec based multicast and justify the reasons why we choose IPsec in this work.

The IPsec protocol suite is a mature and sophisticated solution for secure data communication and key management. As mentioned in [11, 85], IPsec and IKE have been implemented on nearly all modern operating systems and used widely by security communities. There are a number of third-party interfaces and toolkits to configure and manage IPsec. IPsec has undergone a degree of formal analysis demonstrating that it preserves a variety of security properties.[3].

IPsec-based multicast is able to support critical multicast applications across wide area networks like Phasor Measurement Unit (PMU) applications [56] and GOOSE messages between substations or between substations and control centers [34, 37]. This enables the same protocols to be used for both local security and security over multiple networks and avoids complicated encapsulation or tunneling (for example, L2TP is used in IEC 61850-90-1 for inter-substation GOOSE).

Additionally, our experiments in Chapter 7 show that IPsec multicast is adequately scalable and efficient, maintaining latencies well below the 4ms target for substations of increasing sizes.

One debate of deploying security protocols like IPsec in process control systems is whether micro-processor based control devices are competent to cryptography computation. Actually, up-to-date IEDs, especially those IEC 61850 enabled IEDs, are full-fledged systems with strong computing and networking capabilities. They can get steady power support and reliable network connections by strengthened network devices. Therefore, it is not a big challenge for this class of control systems to utilize sophisticated security technologies like IPsec.

[3]Actually IKEv1 was shown to have significant vulnerabilities, which were then mitigated in IKEv2

6.6.2 Individual Source Authentication

In the architecture, individual source authentication is achieved by a cross-layer source authentication approach. According to security extended configuration files, GPE is able to determine all network layer parameters of multicast applications, including the source IP address of publisher P, the source IP addresses of all subscriber S_i and the group multicast address. By configuring GIKE and running the group key exchange protocol according to the above information, the architecture sets up GSAs of each group member by unique source IP addresses for each group security association. Because main stream IPsec implementations enforce that the source IP addresses of outgoing IPsec packets must match the local host IP address and IPsec SP selectors, it guarantees each subscriber S_i only delivers multicast packets from the authentic publisher P. In the case that one encrypted publisher hosts multiple applications, we assign a unique multicast address for each application. Considering the fact that there are usually tens or hundreds of multicast applications in control networks like substation networks, this approach is competent to cover most scenarios. Thus, individual source authentication is specified in the application layer and implemented in the network layer.

6.6.3 DoS Protection

Relying on IPsec's authentication features, IPsec multicast is resilient to some DoS attacks from the transport layer. For example, because TCP control packets are authenticated in IPsec, DoS attacks that depend on the use of TCP control messages can be mitigated. By enforcing the policies in GSPD, group members are able to discard some trivial flooded data packets. Actually IPsec provides protection at network layer and all layers above it.

DoS attack is also a crucial threat to IPsec/IKE and multicast at the network layer [33, 55, 19]. Academic and industrial communities already propose a number of DoS-resilient solutions [24, 1, 75, 49] to mitigate DoS attacks in these areas, including replacing DoS-vulnerable IKEv1 with IKEv2, which does not perform much processing until it determines if the requester can participate in a round trip communication. The GDOI is a potential DoS attack target since the group key protocol is still based on IKEv1. IETF is working on a new group key management protocol based on IKEv2 [63], which partially addresses the problem.

There is an additional concern about duplicated GOOSE messages. Because GOOSE does not require acknowledgements from the recipient, the sender repeats sending duplicated GOOSE messages to achieve the reliability. If each duplicated message need to be signed/verified or encrypted/decrypted, the computation overhead on the senders and the recipients would be a challenge. This problem needs to be explored further.

6.7 Case Study: TVA Bradley IEC 61850 Substation

Based on an experimental configuration for TVA Bradley IEC 61850 substation, We develop a case study to demonstrate the usability of SecureSCL. It shows how SecureSCL derives group associations, sets up IPsec multicast tunnels, and implements timing critical multicast in a substation network.

The Bradley 500-kV substation is the first fully automated, multi-vendor project in the United States to implement the full suite of IEC 61850 communications [68]. It integrates nearly 50 IEDs from three vendors. 34 IEDs are involved in GOOSE communications. More than 40 multicast groups transmit more than 400 data objects. It is a typical IEC 61850 deployment in a transmission power substation.

6.7.1 Substation Configuration in SecureSCL

Our case study is based on a trimmed and revised Bradley configuration. It is actually the practical formation for the motivation example in Section 4.1. Figure 6.10 shows the network topology of the case study. In the

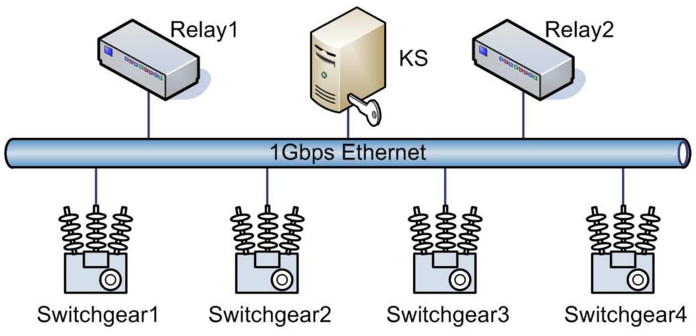


Figure 6.10: Case Study: A Portion of TVA Bradley Substation

rest of this section, we will introduce the case study based on the Appendix.

Six IEDs are connected by a 1Gbps Ethernet LAN (Line 14 and 15). Two of them are protective relays (Line 90 and 121) and the rest are switchgears (Line 154, 185, 218 and 247). Each relay has a logical device PROT (protection, Line 94 through Line 114 for Relay1)[4], which consists of two logical nodes: PDIS repre-

[4]We take Relay1 as an example to illustrate protective relays.

senting the distance protection scheme (Line 112), and PTRC representing protection trip conditioning (Line 111). The combination of PDIS's logical node Op and PTRC's logical node Tr is usually the main part of a TRIP command (Line 97 through 100 for Relay1). In the case study, we only put the data attribute general in GOOSE. In a real system, some meta information like the time-stamp and the quality are transmitted too. Each relay has a logical node GGIO (Line 113), whose data objects can be mapped to a variety of physical parameters like voltage volts of a transformer. In this case, the GGIO on Relay1 has two data objects Ind11 and Ind12 representing two indicators for power grid status (Line 102 through 107). The attribute stVal is the value of the status data. Each relay sets up two GOOSE messages, *i.e.* two multicast groups: one for the TRIP command (Line 108) and the other for the status update (Line 109). The payloads of the messages are defined in the elements DataSet (Line 96 through 107) and the application layer description of the multicast is specified in GSE element. The network configuration of the relays and their GOOSE messages are defined in the corresponding ConnectedAP elements (Line 30 through 42). The ConnectedAP also defines the multicast addresses for the publisher (Line 37 and 40).

Each switchgear has a logical device CTRL (Line 158 through 178 for Switchgear1 [5] which consists of a single logical node XCBR representing circuit breaker (Line 160). Each XCBR monitors a TRIP command and a status update message. The data requirements are defined in Inputs and ExtRef elements (Line 161 through 170). Their network information, like IP addresses, is also defined in corresponding ConnectedAP elements (Line 60 through 65). All IEDs' security credentials like X509 certificates are specified by the element KeyInfo (Line 180 througth 180) and put in the AccessPoint element.

Both the relays and the switchgears have some data objects or attributes, which are not published by GOOSE. For example, each switchgear has a data object Pos indicating the switch position (Line 171 through 176), but this data is not transmitted.

A group controller and key server KS is introduced to the system. It is defined in the new element GCKS (Line 16 through 29), including network parameters and credentials.

This SecureSCL based SCD file is loaded by all emulated IEDs and the group controller in the tested. Multicast model is derived from the configuration and transformed to authorization policies or the configuration of the underlying group key exchange protocol (GDOI). For IEDs, they obtain the group controller's information like the certificate, and run the group key exchange protocol to set up GSPD and GSAD for traffic regulation and security.

[5]We take Switchgear1 as an example to illustrate switchgears

6.7.2 Discussion of Configuration Anomaly Detection Algorithms

For an anomaly detection algorithm, one of the most important evaluation criteria is the rate of false positive or false negative. To get this data, we need a number of benchmark SCL files, especially those from real IEC 61850 substations. Unfortunately, IEC 61850 is a fairly new specification for substation automation. It is very hard to get sufficient benchmarks. The SCD file of TVA Bradley substation is already used in the production system. It has undergone extensive investigation and analysis and a number of configuration mistakes have already been detected by manual check. So, we cannot provide solid data about the false positive or false negative rate at the moment. However, we do insert artificial mistakes in our benchmark SCD file. SecureSCL detects all of these anomalies successfully.

Another issue is the scalability performance of the algorithms. To test the performance of SecureSCL and the algorithms, we deploy SecureSCL on a PC running Fefora 7 with Intel Core 2 Duo E6600 @ 2.4GHz and 2GB memory, and measure the latency of using each detection algorithm to check the whole multicast model derived from the benchmark SCD file. For each algorithm, we run 1000 times and get the average latency. Since the system usually needs warming up executions, we also measure the latency for the first run. The result in Table 6.1 looks encouraging. In these 6 objects and 19 data objects benchmark system, the time used to check the whole model is less than 200 micro seconds. Considering the time complexity analysis in Section 5.2, we can extrapolate that the algorithms is capable of handling regular power substation multicast systems efficiently.

Anomalies	Ownership Anomaly	Publication Redundancy	Source Anomaly	Data Dissatisfaction
First Run (us)	39.7	45.4	38.4	54.3
Ave.(us)	28.3	38.2	15.9	59.3

Table 6.1: Performance of Anomaly Detection Algorithms

In summary, SecureSCL is practical and efficient for secure multicast configuration and initialization for power grid communications. It can detect multicast configuration anomalies with the tolerable time latency. The detection algorithms' efficiency in terms of the rate of false positive or false negative need to explore further. In the next chapter, we will study the latency performance of IPsec based multicast, which is another corner stone of the reference architecture.

Chapter 7

Performance Analysis of IPsec-based Multicast

In this chapter, we test the idea of using IPsec to secure multicast for medium scale LANs like power substation networks. We discuss three candidate schemes for IPsec-based multicast: full-graph, hub-and-spokes and native IPsec multicast, focusing on the issues of latency, communication overhead, and management burdens. Based on a commodity implementation of IPsec, we design experiments to compare the two popular schemes. The result shows "native IPsec multicast" is quite scalable and efficient, maintaining latencies well below the 4ms target for substation networks of increasing size.

7.1 IPsec-based Multicast Schemes

In this section, we compare the three IPsec based multicast schemes by studying their features for a general multicast group. We mainly study the overhead or latency difference between different schemes.

We assume the size of the multicast group is n. All group members including additional network devices have the same computation capacities and the same cryptographic algorithms are chosen for the study. Finally, we assume there is only one sender in a single session and the rest of group members are recipients only.

7.1.1 Full Graph Scheme

IPsec is originally designed as a suite of point-to-point and pairwise security protocols. A straightforward solution for secure multicast is the *full graph scheme* where tunnels are set up between each pair of group members by running IKEv1 or IKEv2. Figure 7.1(a) shows the illustrative diagram of the full graph scheme.

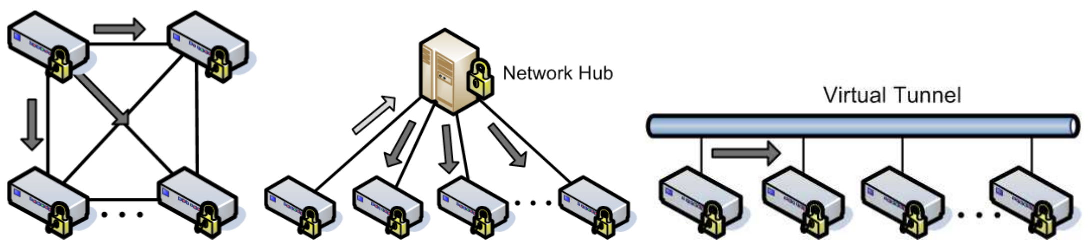
This solution requires $n\cdot(n-1)/2$ "tunnels". Each group member maintains $n-1$ pairs of security credentials, like IPsec SAs and SPs, for the $n-1$ IPsec tunnels to other members. To multicast, or actually broadcast a message within the group, $n-1$ duplicated messages are sent, one for each recipient. Because it is actually a multi-unicast system rather than a true multicast system, the sender sends the message sequentially and all recipients deliver the message in the order of the time when each unicast message is sent. That is, the recipients cannot receive the messages simultaneously. Thus, the extra "intrinsic" delay will be introduced and the delivery latency to the last recipient is the longest. Since in a substation network, every recipient may play an important role in power protection, such a delay is a serious risk to the system. Therefore, the full graph scheme cannot guarantee the latency requirement for critical messages in power grid networks.

7.1.2 Hub-and-Spokes Scheme

The *hub-and-spokes scheme* is a classical solution supporting point-to-point or hop-by-hop security tunnels [69, 13, 12]. It take advantages of a network hub ("hub" in short. We call it "network hub" to emphasize it works at the network layer) which is connected to all group members via IPsec tunnels. Messages are routed to the network hub through the "upstream" tunnel and then relayed to the recipients through the "downstream" tunnels. Figure 7.1(b) shows the illustrative diagram of the hub-and-spokes scheme.

Like the full graph scheme where only point-to-point IPsec tunnels are used, the hub-and-spokes scheme also transforms a multicast message into multiple unicast messages. The sender replicates outgoing messages with different destination addresses. By appropriate network and IPsec configuration, all messages are tunneled to the network hub and then forwarded to the corresponding tunnels based on the messages' destination IP addresses. Because the forwarding operation only occurs at the network layer, it is transparent to upper layer applications. The network hub is also unaware of application logic.

The scheme requires n tunnels. The network hub needs to store n pairs of security credentials, while each group member only needs to store one pair of security associations with the network hub. Like the full



(a) Full graph (b) Hub-and-spokes (c) Native multicast

Figure 7.1: IPsec-based Multicast Schemes

graph scheme, $n-1$ duplicated messages are required to transmit one message. The extra latency due to the sequential message delivery is also introduced. Because all messages are forwarded by the network hub, one more hop is added. Although centralized topology and administration can better support desired group partition and audit, it is a challenge to scalability. Our experiments show the performance is downgraded as the network scale increases.

Copy-and-Forward is a hybrid approach between a full graph scheme and a hub-and-spokes scheme in which a root node reproduces a single message from the sender and relays duplicates to a collection of other nodes which further distribute the message by making a copy and forwarding it according to a suitable scheme such as a spanning tree. While this trades off between the advantages and disadvantages of the two extreme cases, it has the disadvantage of being relatively complicated and possibly adding to latencies because of multi-hop deliveries. This method is probably not practical for substation networks in short term and we do not consider its performance here.

7.1.3 Native IPsec Multicast

IPsec implementations on most off-the-shelf operating systems are able to protect multicast packets directly. If the destination IP of an IPsec packet is a multicast address, a host, which joins the multicast group and has appropriate SAs and SPs, is able to deliver the packet. A native IP multicast encapsulation avoids the packet replication that occurs in the previous two schemes. The most important advantage is that all recipients can receive the message nearly simultaneously and no extra latency is introduced since it is a true multicast. This feature is significant for timing critical messages. Figure 7.1(c) shows the illustrative diagram of the native IPsec multicast scheme.

In contrast to the previous two schemes, the downside of the native IPsec multicast is the complicated group key exchange. Because the group keys and credentials are required to share among group members, the pairwise IKEv1 and IKEv2 protocols, which are designed for point-to-point security, cannot be used directly. Although a number of group key management protocols are already proposed by the academic community [65, 14, 81, 6, 4], they are too complex to deploy. In this work, we take one of the most convenient solutions, the GDOI (see Section 2.4.2) and deploy a group controller and key server for a substation network.

The introduction of the key server does not change the features of data flow. In this scheme all hosts share one *virtual tunnel*. Since each IPsec packet is a multicast packet, no duplication is needed and the recipients can deliver messages directly and simultaneously. Due to the restriction of existing IPsec implementations, each group member needs to maintain one set of security credentials for outgoing packets and $n-1$ sets of security credentials for incoming packets, *i.e.* 1 key for outgoing packets and $n-1$ keys for incoming packets. The details of IPsec configuration for native multicast are shown in Section 7.3.1.

Table 7.1 summarizes the features of the three schemes. We can see the sophisticated native multicast scheme has the best performance in terms of the bandwidth usage and the transmission latency. The experiment results in Section 7.4 prove that.

Scheme	Keys on each member	Duplicated messages	Delay	Hops	Aux. devices	Group key protocol
Full graph	$2 \cdot (n-1)$	$n-1$	Yes	1	-	No
Hub-and-spokes	2	$n-1$	Yes	2	Hub	No
Native multicast	n	1	No	1	Key server	Yes

Table 7.1: Comparison of IPsec based Multicast Schemes

7.2 Process Control Emulation System

To compare the performance of the hub-and-spokes scheme and the native multicast scheme, we design a Process Control Emulation System (PCES) for emulating substation multicast messages. The messages behave like GOOSE messages in the network layer and we call them *GOOSE-like* messages.

PCES is actually an application layer emulator, as well as a performance testing tool. It encapsulates messages in UDP unicast or multicast packets with IPsec and measures round trip latencies. The manipulation of IPsec SAD and SPD is supported by the testbed (see Section 7.3.1) and transparent to PCES. PCES has two versions: PCES-HS (**H**ub-and-**S**pokes) designed for the hub-and-spokes scheme and PCES-MC (**M**ulti**C**ast) for the native IPsec multicast. Both of them are written in C/C++ and deployed on Linux platforms.

7.2.1 PCES for Hub-and-Spokes

PCES-HS transforms multicast messages into multiple unicast messages. Each host in PCSES-HS is assigned an ID. The sender in PCES-HS sends requests one by one in the ascending order of recipients IDs.

Each request is wrapped in UDP packets with individual unicast IP addresses to all recipients. The timestamp of each request is recorded when it is being sent. After receiving the request, the recipients will respond with an acknowledgement message immediately. Upon receiving the acknowledgement, the sender calculates the round trip latency to each recipient based on the recorded timestamp and the current time. PCES-HS is not deployed on the network hub, which only maintains IPsec tunnels and forwards packets.

Because PCES-HS need send requests and process acknowledgements concurrently during the early stage of each testing run, extra latency and uncertainty are expected. Before the sender finishes sending all requests, it may already receive the acknowledgements from the recipients which receive requests earlier. This will keep the sender in busy status and some responses might be queued, which would cause extra latency. We will discuss this problem in depth based on the experiment results in Section 7.4.2.

7.2.2 PCES for Native Multicast

PCES-MC is designed for native IPsec multicast. A sender only sends one copy of request in UDP with a multicast destination IP. All recipients should be able to receive the request nearly simultaneously. However, because the sender only can process one acknowledgement at one time and some acknowledgements may be queued just like it occurs in PCES-HS; it is hard to accurately measure the round trip latency for each recipient.

To address the problem, PCES-MC does not have all recipients acknowledge the request. Given that all hosts have same computation capacity and connected with same bandwidth links, we assume they will receive the request simultaneously and respond at the same time. The duration, from the time when the sender sends the request to the time when all recipients receive the request and get ready to respond, is just the *application-to-application communication time* defined in IEEE1646 [72]. PCES-MC picks only one recipient randomly to respond to the request, and only records the timestamp for this acknowledgement. All remaining recipients will discard the request and keep silent. Thus the sender only need process one acknowledgement for one session and will not be overwhelmed. Considering the minor difference between group members, the test will be repeated many times (1000 times per round) to measure the latencies from different recipients. This sampled round trip latency measurement method can collect precise latency data and eliminate uncertain delay. We expect the latency will not increase as the network scale grows and the standard deviation is small.

7.3 Testbed Setup

7.3.1 IPsec GSA/GSP Configuration

Before setting up the testbed, we first show how to configure IPSec Security Association (SA) and Security Policy (SP) for multicast communications. We call such SA and SP Group Security Association (GSA) and Group Security Policy (GSP) respectively.

Figure 7.2 shows two GSA examples on a host with the IP address 10.1.1.4. Line 1 through 5 defines a GSA for outgoing packets. The original packet will be encapsulated in tunnel mode using the ESP protocol with a source IP of the local host and a multicast destination IP of 224.0.0.4. The packet will be authenticated using HMAC-SHA1 and encrypted using AES(CBC). The keys are defined in Line 4 and Line 5. Line 6 through 10 defines a GSA for incoming multicast packets from the host 10.1.1.3. It uses the same mode and protocol as the GSA for outgoing packets. The keys used for authentication verification and decryption are defined in Line 9 and Line 10.

```
[1] src 10.1.1.4 dst 224.0.0.4
[2] proto esp spi 0x06002999
[3] reqid 0 mode tunnel
[4] auth hmac(sha1) 0x0d...393
[5] enc cbc(aes) 0x68...7af
...
[6] src 10.1.1.3 dst 224.0.0.4
[7] proto esp spi 0x06001999
[8] reqid 0 mode tunnel
[9] auth hmac(sha1) 0x47...953
[10] enc cbc(aes) 0xfb...b86
...
```

Figure 7.2: Group Security Associations

```
[1] src 10.1.1.4/32 dst 224.0.0.4/32
[2] dir out priority 0 ptype main
[3] tmpl src 10.1.1.4 dst 224.0.0.4
[4] proto esp spi 0x06002999
[5] mode tunnel
...
[6] src 10.1.1.3/32 dst 224.0.0.4/32
[7] dir in priority 2080
[8] tmpl src 10.1.1.3 dst 224.0.0.4
[9] proto esp spi 0x06001999
[10] mode tunnel
...
```

Figure 7.3: Group Security Policies

Figure 7.3 shows two GSP examples on the same host. Line 1 through 5 define a policy for outgoing packets whose source IP is the local host and the destination IP is 224.0.0.4. Such packets will be encapsulated by the GSA whose ID is specified from Line 3 through 5, *i.e.* the first GSA in Figure 7.2. Line 6 through 10 defines a policy for incoming packets whose source IP is 10.1.1.3 and the destination IP is 224.0.0.4. The packets will be de-capsulated by the GSA whose ID is specified from Line 8 through 10, *i.e.* the second GSA in Figure 7.2.

We can see that each member has a set of keys used to encrypt and sign the outgoing packets and share the keys with other members. The key distribution is completed by the key server. It is possible to use a same key for all members. In reality, most main stream IPsec implementations enforce that the source IP address of an outgoing IPsec packet must be same as the local host IP address and IPsec SP selectors. Therefore individual source authentication can be achieved even a same key is used within the whole group.

There are a couple of ways to set up GSA and GSP manually or automatically. Linux systems provide a built-in command-line tool named setkey [43] for manually manipulating the IPsec SAD and SPD. Some third-party IPsec key management tools like iproute2 [42] also provide command-line tools for manual manipulations. These tools may use different underlying library interfaces. Another approach is to run group key management tools like the GDOI [10]. In our testbed for IPsec performance testing, we choose iproute2.

7.3.2 Testbed

We deploy PCES on the DETER Testbed [18], a public facility for medium-scale repeatable experiments in computer security. The testbed consists of PCs running Ubuntu 8.04 with Linux kernel version 2.6.24 and strongSwan [71] version 4.3.0, a third-party IPsec configuration tool. Xeon Quad 3.00GHz PCs with different size caches are assigned by the DETER Testbed administrative system. Tests show the cache size does not affect the performance too much for our experiments.

For the experiments of the hub-and-spokes scheme, SPDs and SADs on all hosts including the network hub are initialized by strongSwan's IKE tool. The tool runs IKEv1 between each host and the network hub automatically when the emulated network is being initialized. Each PCES-HS instance assumes it is talking to the destinations directly, though all packets are actually forwarded via the network hub. Considering integrity is the main concern in substation networks, we only use SHA1 for ESP, no encryption is applied.

The configuration of native IPsec multicast scheme, *i.e.* setting up GSAs and GSPs, are supported by a third-party IPsec configuration tool iproute2 [42]. To see the degree that the encryption computation affects the performance, we use both HMAC-SHA1 and AES for ESP.

Using DETER testbed's GUI tools, NS2/TCL based script tools and shell scripts, we specify the network topology, install and configure the operating system, IPsec, strongSwan, PCES systems and credentials on each host automatically when the assigned hosts are booting. Figure 7.4 shows the network topology of the 8-host experiment for PCES-HS, which is created by the DETER's GUI tool for the network topology

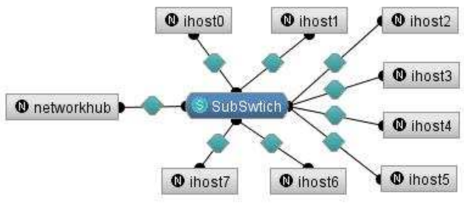


Figure 7.4: 8-Host Experiment Topology in PCES-HS

design. All hosts including the network hub are connected by a 1Gbps Ethernet switch. One challenge in the testbed initialization is the testbed synchronization, *i.e.* to start the experiments after all hosts finishes initialization. For example, in PCES-HS each group member cannot run IKE until other hosts have finished the installation and the network hub has started IPsec service. In PCES-MC, the experiment only can start until all hosts finish the setup of GSA and GSP. The DETER testbed provides a mechanism called *barrier* for synchronization. It allows programs to wait at a specific point, and then all proceed at once. In PCES-HS, we set the network hub as the synchronization server to coordinate all hosts and guarantee the configuration process runs as expected. In PCES-MC, we choose a random group member as the synchronization server.

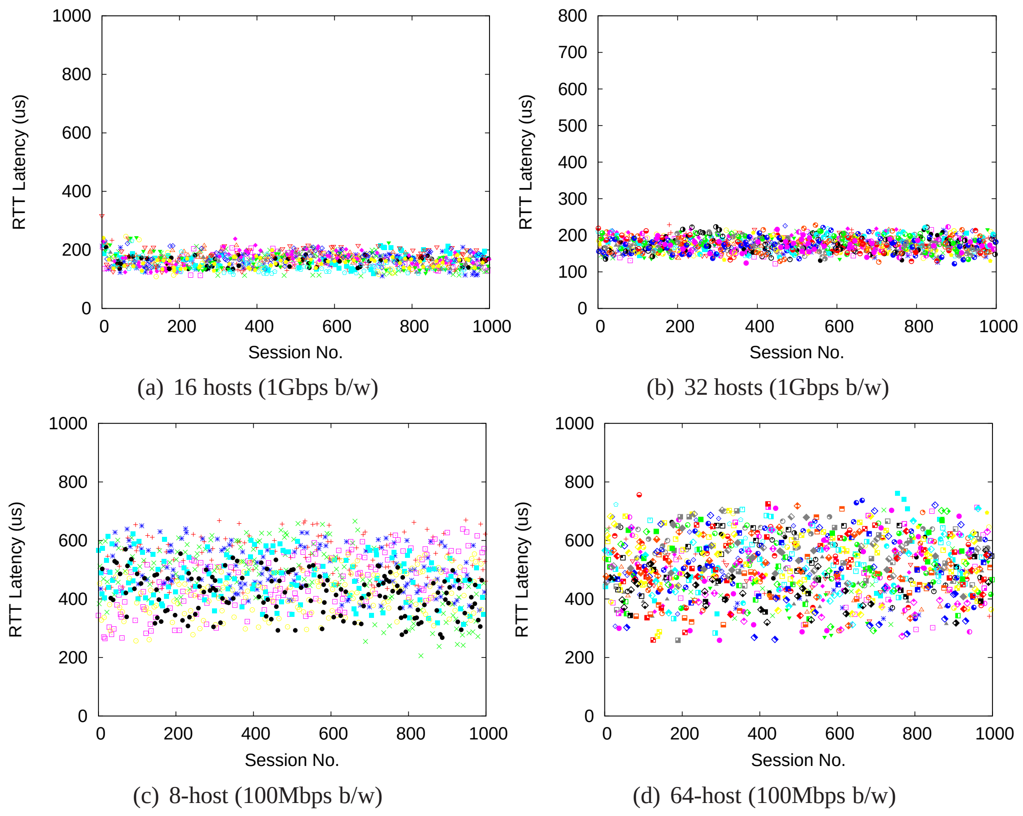
Although the DETER testbed usually can provide more than 100 free PCs at one time, not all of them are connected in a same 1Gbps Ethernet LAN. Due to some reasons, when we were running the experiments of PCES-MC, we could not obtain sufficient 1Gbps Ethernet switches for 64-host experiments and all hosts are not located at a same DETER testbed site. So we run the 64-host experiments of PCES-MC by 100Mbps LANs. Fortunately, the experiment result is still positive to our research. But when we were running PCES-HS experiments a few months before the PCES-MC experiments, the testbed did allocate sufficient resources for us. So, all the experiments results of PCES-HS are based on 1Gbps LANs.

7.4 Results and Discussion

7.4.1 Scalable Native IPsec Multicast

To test the latencies under different network scales for native multicast, we create the experiments for the network sizes of 4, 8, 16, 32 and 64 hosts respectively. In each experiment, we run PCES-MC 8 rounds (4 rounds in case the network size is 4). In each round, we randomly pick one sender and have others listen and acknowledge. The sender multicasts requests periodically (1000 sessions) in a 140-byte UDP packet and recipients respond an acknowledgement with the same payload size.

Figure 7.5(a) and Figure 7.5(b) show the latencies of native IPsec multicast from a sending host with



(a) 16 hosts (1Gbps b/w)
(b) 32 hosts (1Gbps b/w)
(c) 8-host (100Mbps b/w)
(d) 64-host (100Mbps b/w)

Figure 7.5: Performance of Native IPsec Multicast

the network sizes of 16 hosts and 32 hosts in a 1Gbps switched Ethernet. The X-axis represents Session No. of each round experiment; the Y-axis represents round trip latency in the unit of microsecond. The dots indicate the latencies from different recipients, which are represented by the dots in different shapes and colors. Because we cannot gain sufficient 1Gbps switches in a same LAN for the 64-host experiment due to resource limits in the DETER testbed, we have the test, as well as an 8-host experiment for comparison, on 100Mbps LANs (see Figure 7.5(c) and Figure 7.5(d)). This set of data are shown in the 8* and 64* columns of Table 7.2. A box-whisker graph using the same data is shown in Figure 7.6.

According to the data, in a 1Gbps switched Ethernet, when the network size increases from 4 hosts to 32 hosts, most latency are less than 200us and the longest latency is less than 300us. The average latency is less than 200us and the standard deviation is between 20us to 25us (see Table 7.2). The result also shows the bandwidth affects the latency. The average latencies for 8-host and 64-host scenarios are 466us and 495us

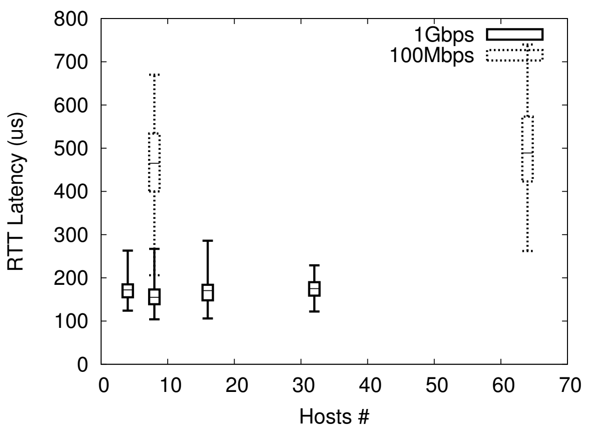


Figure 7.6: Performance of Native IPsec Multicast: A Combined View

Network size	4	8	16	32	8*	64*
Ave.(us)	171	156	169	174	466	495
Std.(us)	22.4	22.1	25.9	20.8	92.3	102

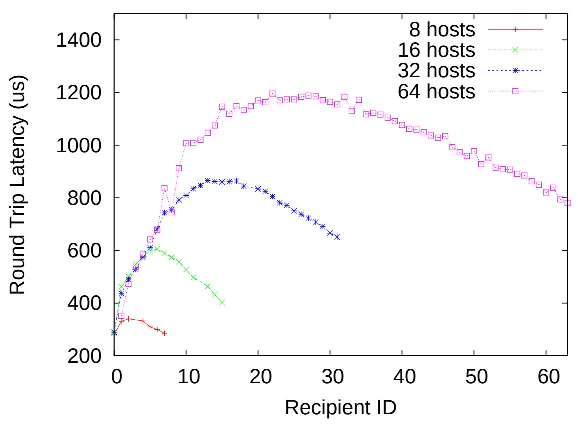
Table 7.2: Avg. & Stdv. of Round Trip Latency for Native Multicast Scheme

respectively and the standard deviations are 92.3us and 102us, which are much larger than the numbers in 1Gbps LANs. The data show that native IPsec multicast is competent in fast packets transmission even the network bandwidth is limited. As the network size increases, its performance is not degraded remarkably. In general, the native IPsec multicast is quite scalable and efficient, maintaining latencies well below the 4ms target for substation networks of increasing scales.

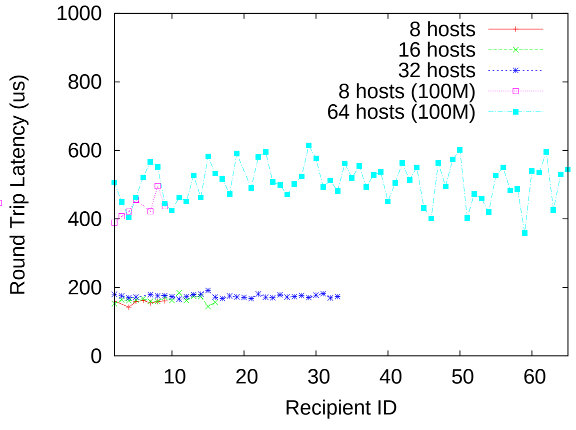
7.4.2 Analysis of Hub-and-Spokes Scheme

We design the experiments to analyze the performance of the hub-and-spokes multicast scheme. First of all we assign each host an ID. Then we also create the experiments for the network sizes of 8, 16, 32 and 64 hosts respectively (Fortunately we obtain sufficient resources from the DETER lab for 64-host experiment). Like the experiments in PCES-MC, we randomly pick one sender for each experiment and have others listen and acknowledge. . The sender sends requests periodically (500 sessions) in 140-byte UDP multicast packets. The recipients respond an acknowledgement with the same payload size.

We calculate the average round trip latency to each recipient and plot the data on Figure 7.7(a). The X-axis represents recipients' IDs; the Y-axis represents the latency in the unit of microsecond. Each dot



(a) Hub-and-spokes Scheme



(b) Native IPsec Multicast

Figure 7.7: Average Round Trip Latency of Hub-and-spokes Scheme and Native IPsec Multicast

represents the average round trip latency from the sender to the recipient whose ID is indicated by the X-axis. The dots for one experiment with the certain number of host are connected by a line, namely *latency line*.

Each latency line represents an experiment with certain network scale. We can see the lines representing larger network scales are much "higher" than the lines representing smaller network scales, *i.e.* the latencies increase rapidly as the network size grows. The largest average round trip latency increases from around 300us to 1200us as the network scale increases from 8 hosts to 64 hosts.

On the other hand, within a single experiment, the latencies from different recipients differ quite much. The latency lines appear as the curves, which rise rapidly as the recipient ID increases and begins decreasing at a particular point. This problem is caused by the hub-and-spokes scheme, as well as the experiment methods used in PCES-HS. We discuss the problem in details below.

As mentioned in Section 7.2.1, in PCES-HS the sender always sends request messages in an ascending order of recipients' IDs. Therefore, the smaller ID a recipient has, the earlier it receives the request and acknowledges. The sender could be overwhelmed by the incoming acknowledgements before it finishes sending requests. Extra latency and uncertainty are introduced at that time. So the curve rises rapidly just after the experiment starts, *i.e.* the calculated latencies to the recipients which send acknowledgement early become quite long. At a particular point, the sender sends out all requests and start processing acknowledgements only. At the moment, the latencies of the subsequent sessions decrease. But because some packets may have stayed in the queue for a while, the latencies are still large.

We find, in an experiment, the latency and standard deviation from the first recipient are always the smallest (See Table 7.3) because the sender has not been overwhelmed by acknowledgements at that moment

Network size	8	16	32	64
Ave.(us)	280	285	289	351
Std.(us)	33.4	56.9	119	473

Table 7.3: Avg. & Stdv. of Round Trip Latency for Hub-and-spokes Scheme

and the first acknowledgement gets time stamped with least interference. We can see even in the best case, the performance of the hub-and-spokes scheme is worse than average performance of the native multicast scheme. Actually, as the network scale increases, the standard deviations of the latencies become larger and larger. In some experiments, we find the value for those recipients with large IDs increases from 68.56us for 8-host scale to 7248us for 64-host scale. Such fluctuation is not acceptable for power grid communications.

Indeed, because the hub-and-spokes scheme transforms a multicast message to multiple unicast messages, recipients must receive the message in a precedence order. Intuitively, given the network bandwidth and the hosts' capacity, as the network scale grows, this latency will increase proportionally. Given that every host has equal priority for power protection, the last recipient receiving the request very likely misses the time window and cannot react to emergent events timely. Therefore, the hub-and-spokes scheme is not capable of handling timing critical multicast communications in power grid networks.

7.4.3 Comparison of Hub-and-Spokes and Native Multicast Schemes

To compare the latencies of the hub-and-spokes scheme and native multicast scheme, we transform the plots in Figure 7.5 to Figure 7.7(b), plus the experiments of 4-host and 8-host. We also assign each host, calculate the average round trip latency to each recipient, and plot the data on the diagram. The X-axis represents recipients' IDs; the Y-axis represents the latency in the unit of microsecond. Each dot represents the average round trip latency from the sender to the recipient whose ID is indicated by the X-axis. The dots for one experiment with the certain number of host are connected by a latency line.

The results show the latencies do not increase remarkably as the network scale increases. Although the latency lines fluctuate much more in 100Mbps LANs than in 1Gbps LANs, *i.e.* the standard deviation is larger. They are still in acceptable range.

Based on the experiment results and the above analysis, we conclude that native IPsec multicast is more capable of addressing timing critical multicast. As the network size increases, its performance does not decline remarkably. It is appropriate to raise GOOSE to the network layer for IPsec protection. In general, native IPsec multicast is quite scalable and able to maintain latencies well below the 4ms target for substation networks of increasing sizes.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

The application-aware secure multicast architecture is an efficient solution for multicast applications in power grid systems. By analyzing derived multicast models and checking data dependencies based on functional configurations, it automates group management and minimizes errors due to manual configurations. The architecture integrates security information with functional configurations and takes advantage of off-the-shelf security technologies. IPsec is a promising solution for secure multicast in power grid systems. It is capable of transmitting timing critical messages with the guarantees of integrity and confidentiality. Our experiments show it can meet the target latency of 4ms benchmark used for power substations. The performance is not downgraded remarkably as the network size grows.

This work provides a cross-layer approach of automatically self-generated group configuration for power grid communications, addressing key concerns of both system implementation and conformance analysis. The proposed multicast model and verification mechanism can be extended for generic secure communication configurations. On the other hand, the prototype system SecureSCL has a potential of being developed into a realistic application for power substations.

8.2 Future Work

The research already completed on application-aware derived group multicast suggests a rich field of further research with important benefits. Future work in this area directly motivated by our work include:

- Dynamic group management. Current data dependency analysis and the multicast formal model rely on static configuration files of power grid system. The multicast model can be extended to support dynamic environment where group members join or leave the group frequently. Data dependency analysis could be based on the change of data flow in the system. Such dynamic multicast model can be used in various areas, like pervasive computing.

- Cross-network or inter-substation network multicast communication and configuration. IEC 61850 is designed for local area network only. An extension for multicast between substations or between substations and control centers is promising solution for power grid systems. The collaborated and wide area network multicast configuration would be interesting topic too.

Appendix

Case Study: Extended Substation Configuration

We present here the full SCD file used for the case study of the TVA Bradley IEC 61850 Substation described in Section 6.7.

```
<?xml version="1.0" encoding="UTF-8"?>
<SCL xsi:schemaLocation="http://www.iec.ch/61850/2003/SCL SCL.xsd"
  xmlns="http://www.iec.ch/61850/2003/SCL"
  xmlns:sscl="http://seclab.illinois.edu/SecureSCL"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:ns="http://www.iec.ch/61850/2003/SCL">

  <Header id="SecureSubstaion-Jun2009" revision="5" version="1">
    <Text>SECURE SUBSTATION</Text>
  </Header>
  <Communication>
    <SubNetwork name="TVACaseStudy" type="SecSubnet">
      <BitRate multiplier="M" unit="b/s">1000</BitRate>
      <sscl:GCKS desc="Group Controller and Key Server" name="KS">
        <Address>
          <P type="IP">192.168.1.2</P>
          <P type="IP-SUBNET">255.255.255.0</P>
          <P type="IP-GATEWAY">192.168.1.1</P>
        </Address>
        <sscl:GIKE>
          <sscl:GroupProtocol>GDOI</sscl:GroupProtocol>
          <sscl:Port>848</sscl:Port>
        </sscl:GIKE>
        <ds:KeyInfo Id="GCKS">
          <ds:X509Data><ds:X509Certificate>MI...</ds:X509Certificate></ds:X509Data>
        </ds:KeyInfo>
      </sscl:GCKS>
      <ConnectedAP apName="apRelay1" desc="Relay1 AP" iedName="Relay1">
      <Address>
        <P type="IP">192.168.1.20</P>
        <P type="IP-SUBNET">255.255.255.0</P>
        <P type="IP-GATEWAY">192.168.1.1</P>
      </Address>
      <GSE cbName="gcbTrip1" ldInst="PROT">
        <Address> <P type="IP">224.0.0.4</P> </Address>
      </GSE>
      <GSE cbName="gcbST1" ldInst="PROT">
        <Address> <P type="IP">224.0.0.5</P> </Address>
      </GSE>
    </ConnectedAP>
    <ConnectedAP apName="apRelay2" desc="Relay2 AP" iedName="Relay2">
      <Address>
        <P type="IP">192.168.1.21</P>
        <P type="IP-SUBNET">255.255.255.0</P>
        <P type="IP-GATEWAY">192.168.1.1</P>
      </Address>
      <GSE cbName="gcbTrip2" ldInst="PROT">
        <Address>
          <P type="IP">224.0.0.6</P>
        </Address>
      </GSE>
      <GSE cbName="gcbST2" ldInst="PROT">
        <Address>
          <P type="IP">224.0.0.7</P>
        </Address>
      </GSE>
    </ConnectedAP>
    <ConnectedAP apName="apSwitchgear1" desc="Switchgear1 AP" iedName="Switchgear1">
      <Address>
        <P type="IP">192.168.1.22</P>
        <P type="IP-SUBNET">255.255.255.0</P>
        <P type="IP-GATEWAY">192.168.1.1</P>
      </Address>
    </ConnectedAP>
    <ConnectedAP apName="apSwitchgear2" desc="Switchgear2 AP" iedName="Switchgear2">
      <Address>
        <P type="IP">192.168.1.23</P>
        <P type="IP-SUBNET">255.255.255.0</P>
        <P type="IP-GATEWAY">192.168.1.1</P>
      </Address>
      </ConnectedAP>
      <ConnectedAP apName="apSwitchgear3" desc="Switchgear3 AP" iedName="Switchgear3">
        <Address>
          <P type="IP">192.168.1.24</P>
          <P type="IP-SUBNET">255.255.255.0</P>
          <P type="IP-GATEWAY">192.168.1.1</P>
        </Address>
      </ConnectedAP>
      <ConnectedAP apName="apSwitchgear4" desc="Switchgear4 AP" iedName="Switchgear4">
        <Address>
          <P type="IP">192.168.1.25</P>
          <P type="IP-SUBNET">255.255.255.0</P>
          <P type="IP-GATEWAY">192.168.1.1</P>
        </Address>
      </ConnectedAP>
    </SubNetwork>
  </Communication>
  <IED desc="Protective Relay 1 (P1)" name="Relay1" type="SecureIED">
    <AccessPoint desc="Relay1 AP" name="apRelay1">
      <Server>
        <Authentication certificate="true" none="false" strong="true" />
        <LDevice inst="PROT">
          <LN0 inst="" lnClass="LLN0" lnType="RELAY1_LLN0_Type">
            <DataSet name="dsTrip1">
              <FCDA daName="general" doName="Tr" fc="ST" ldInst="PROT"
                lnClass="PTRC" lnInst="1" />
              <FCDA daName="general" doName="Op" fc="ST" ldInst="PROT"
                lnClass="PDIS" lnInst="1" />
            </DataSet>
            <DataSet name="dsStatus1">
              <FCDA daName="stVal" doName="Ind11" fc="ST" ldInst="PROT"
                lnClass="GGIO" lnInst="1" />
              <FCDA daName="stVal" doName="Ind12" fc="ST" ldInst="PROT"
                lnClass="GGIO" lnInst="1" />
            </DataSet>
            <GSEControl appID="TripGoose" datSet="dsTrip1" name="gcbTrip1"/>
            <GSEControl appID="StatusUpdate" datSet="dsStatus1" name="gcbST1"/>
          </LN0>
          <LN inst="1" lnClass="PTRC" lnType="RELAY1/PTRC" prefix=""></LN>
          <LN inst="1" lnClass="PDIS" lnType="RELAY1/PDIS" prefix=""></LN>
          <LN inst="1" lnClass="GGIO" lnType="RELAY1/GGIO" prefix=""></LN>
        </LDevice>
    </Server>
    <ds:KeyInfo Id="Relay2">
      <ds:X509Data> <ds:X509Certificate>NV...</ds:X509Certificate> </ds:X509Data>
    </ds:KeyInfo>
  </AccessPoint>
</IED>
<IED desc="Protective Relay 2 (P2)" name="Relay2" type="SecureIED">
  <AccessPoint desc="Relay2 AP" name="apRelay2">
    <Server>
      <Authentication certificate="true" none="false" strong="true" />
      <LDevice inst="PROT">
        <LN0 inst="" lnClass="LLN0" lnType="RELAY2/LLN0">
          <DataSet name="dsTrip2">
            <FCDA daName="general" doName="Tr" fc="ST" ldInst="PROT"
              lnClass="PTRC" lnInst="1" />
            <FCDA daName="general" doName="Op" fc="ST" ldInst="PROT"
              lnClass="PDIS" lnInst="1" />
          </DataSet>
          <DataSet name="dsStatus2">
            <FCDA daName="stVal" doName="Ind21" fc="ST" ldInst="PROT"
              lnClass="GGIO" lnInst="1" />
            <FCDA daName="stVal" doName="Ind22" fc="ST" ldInst="PROT"
              lnClass="GGIO" lnInst="1" />
            <FCDA daName="stVal" doName="Ind23" fc="ST" ldInst="PROT"
              lnClass="GGIO" lnInst="1" />
          </DataSet>
          <GSEControl appID="TripGoose" datSet="dsTrip2" name="gcbTrip2"/>
          <GSEControl appID="StatusUpdate" datSet="dsStatus2" name="gcbST2"/>
        </LN0>
        <LN inst="1" lnClass="PTRC" lnType="RELAY2/PTRC" prefix=""></LN>
        <LN inst="1" lnClass="PDIS" lnType="RELAY2/PDIS" prefix=""></LN>
        <LN inst="1" lnClass="GGIO" lnType="RELAY2/GGIO" prefix=""></LN>
      </LDevice>
    </Server>
    <ds:KeyInfo Id="Relay2">
      <ds:X509Data><ds:X509Certificate>UU...</ds:X509Certificate></ds:X509Data>
    </ds:KeyInfo>
  </AccessPoint>
</IED>
<IED desc="Switchgear1 (S1)" name="Switchgear1" type="SecureIED">
  <AccessPoint desc="Switchgear1 AP" name="apSwitchgear1">
    <Server>
```

```
        <Authentication certificate="true" none="false" strong="true" />
        <LDevice inst="CTRL">
          <LN0 desc="Switchgear1_LLN0" inst="" lnClass="LLN0" lnType="Switchgear_LLN0"></LN0>
          <LN desc="CircuitBreaker" inst="1" lnClass="XCBR" lnType="SECURE/XCBR">
            <Inputs>
              <ExtRef daName="general" doName="Tr" iedName="Relay1"
                ldInst="PROT" lnClass="PTRC" lnInst="1" />
              <ExtRef daName="general" doName="Op" iedName="Relay1"
                ldInst="PROT" lnClass="PDIS" lnInst="1" />
              <ExtRef daName="stVal" doName="Ind11" iedName="Relay1"
                ldInst="PROT" lnClass="GGIO" lnInst="1" />
              <ExtRef daName="stVal" doName="Ind12" iedName="Relay1"
                ldInst="PROT" lnClass="GGIO" lnInst="1" />
            </Inputs>
            <DOI name="Pos" desc="Position">
              <DAI name="stVal">
                <Val>2</Val>
                <Text>0-intermediate|1-off|2-on|3-bad</Text>
              </DAI>
            </DOI>
          </LN>
        </LDevice>
      </Server>
      <ds:KeyInfo Id="Switchgear1">
        <ds:X509Data><ds:X509Certificate>TI...</ds:X509Certificate></ds:X509Data>
      </ds:KeyInfo>
    </AccessPoint>
  </IED>
  <IED desc="Switchgear2 (S2)" name="Switchgear2" type="SecureIED">
      <AccessPoint desc="Switchgear2 AP" name="apSwitchgear2">
      <Server>
        <Authentication certificate="true" none="false" strong="true" />
        <LDevice inst="CTRL">
          <LN0 desc="Switchgear2_LLN0" inst="" lnClass="LLN0" lnType="Switchgear_LLN0"></LN0>
          <LN desc="CircuitBreaker" inst="1" lnClass="XCBR" lnType="SECURE/XCBR">
            <Inputs>
              <ExtRef daName="general" doName="Tr" iedName="Relay1"
                ldInst="PROT" lnClass="PTRC" lnInst="1" />
              <ExtRef daName="general" doName="Op" iedName="Relay1"
                ldInst="PROT" lnClass="PDIS" lnInst="1" />
              <ExtRef daName="stVal" doName="Ind21" iedName="Relay2"
                ldInst="PROT" lnClass="GGIO" lnInst="1" />
            <ExtRef daName="stVal" doName="Ind22" iedName="Relay2"
              ldInst="PROT" lnClass="GGIO" lnInst="1" />
            <ExtRef daName="stVal" doName="Ind23" iedName="Relay2"
              ldInst="PROT" lnClass="GGIO" lnInst="1" />
          </Inputs>
            <DOI name="Pos" desc="Position">
            <DAI name="stVal">
              <Val>2</Val>
              <Text>0-intermediate|1-off|2-on|3-bad</Text>
            </DAI>
          </DOI>
        </LN>
      </LDevice>
    </Server>
    <ds:KeyInfo Id="Switchgear2">
      <ds:X509Data><ds:X509Certificate>RE...</ds:X509Certificate></ds:X509Data>
    </ds:KeyInfo>
  </AccessPoint>
</IED>
<IED desc="Switchgear3 (S3)" name="Switchgear3" type="SecureIED">
  <AccessPoint desc="Switchgear3 AP" name="apSwitchgear3">
    <Server>
      <Authentication certificate="true" none="false" strong="true" />
      <LDevice inst="CTRL">
        <LN0 desc="Switchgear3_LLN0" inst="" lnClass="LLN0" lnType="Switchgear_LLN0"></LN0>
        <LN desc="CircuitBreaker" inst="1" lnClass="XCBR" lnType="SECURE/XCBR">
          <Inputs>
            <ExtRef daName="general" doName="Tr" iedName="Relay2"
              ldInst="PROT" lnClass="PTRC" lnInst="1" />
            <ExtRef daName="general" doName="Op" iedName="Relay2"
              ldInst="PROT" lnClass="PDIS" lnInst="1" />
            <ExtRef daName="stVal" doName="Ind11" iedName="Relay1"
              ldInst="PROT" lnClass="GGIO" lnInst="1" />
            <ExtRef daName="stVal" doName="Ind12" iedName="Relay1"
              ldInst="PROT" lnClass="GGIO" lnInst="1" />
          </Inputs>
          <DOI name="Pos" desc="Position">
            <DAI name="stVal">
              <Val>2</Val> <Text>0-intermediate|1-off|2-on|3-bad</Text>
            </DAI>
          </DOI>
        </LN>
        </LDevice>
      </Server>
      <ds:KeyInfo Id="Switchgear3">
        <ds:X509Data><ds:X509Certificate>WQ...</ds:X509Certificate></ds:X509Data>
      </ds:KeyInfo>
    </AccessPoint>
  </IED>
  <IED desc="Switchgear4 (S4)" name="Switchgear4" type="SecureIED">
    <AccessPoint desc="Switchgear4 AP" name="apSwitchgear4">
      <Server>
        <Authentication certificate="true" none="false" strong="true" />
        <LDevice inst="CTRL">
          <LN0 desc="Switchgear4_LLN0" inst="" lnClass="LLN0" lnType="Switchgear_LLN0"></LN0>
          <LN desc="CircuitBreaker" inst="1" lnClass="XCBR" lnType="SECURE/XCBR">
            <Inputs>
              <ExtRef daName="general" doName="Tr" iedName="Relay2"
                ldInst="PROT" lnClass="PTRC" lnInst="1" />
              <ExtRef daName="general" doName="Op" iedName="Relay2"
                ldInst="PROT" lnClass="PDIS" lnInst="1" />
              <ExtRef daName="stVal" doName="Ind21" iedName="Relay2"
                ldInst="PROT" lnClass="GGIO" lnInst="1" />
              <ExtRef daName="stVal" doName="Ind22" iedName="Relay2"
                ldInst="PROT" lnClass="GGIO" lnInst="1" />
              <ExtRef daName="stVal" doName="Ind23" iedName="Relay2"
                ldInst="PROT" lnClass="GGIO" lnInst="1" />
            </Inputs>
            <DOI name="Pos" desc="Position">
              <DAI name="stVal">
                <Val>2</Val> <Text>0-intermediate|1-off|2-on|3-bad</Text>
              </DAI>
            </DOI>
          </LN>
        </LDevice>
      </Server>
      <ds:KeyInfo Id="Switchgear4">
        <ds:X509Data><ds:X509Certificate>WQ...</ds:X509Certificate></ds:X509Data>
      </ds:KeyInfo>
    </AccessPoint>
  </IED>
</SCL>
```

References

[1] W. Aiello, S. M. Bellovin, M. Blaze, R. Canetti, J. Ioannidis, A. D. Keromytis, and O. Reingold. Just fast keying: Key agreement in a hostile internet. *ACM Transactions on Information and System Security (TISSEC)*, 7(2):242–273, 2004.

[2] E. S. Al-Shaer and H. H. Hamed. Design and Implementation of Firewall Policy Advisor Tools. Technical Report Technical Report CTI-techrep0801, School of Computer Science Telecommunications and Information Systems, DePaul University, August 2002.

[3] E. S. Al-Shaer and H. H. Hamed. Discovery of Policy Anomalies in Distributed Firewalls. In *The 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'04)*, March 2004.

[4] Y. Amir, C. Nita-Rotaru, J. Stanton, and G. Tsudik. Secure Spread: An Integrated Architecture for Secure Group Communication. *IEEE Transactions on Dependable and Secure Computing*, 2(3):248 – 261, July 2005.

[5] T. Aurisch and C. Karg. Using the IPsec Architecture for Secure Multicast Communications. In *8th International Command and Control Research and Technology Symposium (ICCRTS)*, Washington, DC, June 2003.

[6] S. Banerjee and B. Bhattacharjee. Scalable Secure Group Communication Over IP Multicast. *Selected Areas in Communications, IEEE Journal on*, 20(8):1511 – 1527, October 2002.

[7] K. Barnes and B. Johnson. Introduction to SCADA: Protection and Vulnerabilities. Technical report, Idaho National Engineering and Environmental Laboratory, Idaho Falls, Idaho, March 2004.

[8] M. Bartel, J. Boyer, B. Fox, B. Lamacchia, and E. Simon. Xml-Signature Syntax and Processing, June 2008. `http://www.w3.org/TR/xmldsig-core/`.

[9] M. Baugher, R. Canetti, L. Dondeti, and F. Lindholm. Multicast Security (MSEC) Group Key Management Architecture. RFC 4046, Apr. 2005.

[10] M. Baugher, B. Weis, T. Hardjono, and H. Harney. RFC3547, The Group Domain of Interpretation, July 2003.

[11] R. Canetti, P. chen Cheng, F. Giraud, D. Pendarakis, J. R. Rao, P. Rohatgi, and D. Saha. An IPsec-based Host Architecture for Secure Internet Multicast. In *Proceedings of the 7th Annual Network and Distributed System Security Symposium (NDSS'00)*, February 2000.

[12] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: Taking Control of The Enterprise. In *SIGCOMM'07*, August 2007.

[13] M. Casado, T. Garfinkel, A. Akella, M. Freedman, D. Boneh, N. McKeown, and S. Shenker. SANE: A Protection Architecture for Enterprise Networks. In *Usenix Security'06*, Vancouver, Canada, August 2006.

[14] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha. Key Management For Secure Internet Multicast Using Boolean Function Minimization Techniques. In *8th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'99)*, March 1999.

[15] N. R. Commission. NRC Information Notice: 2007-15, April 2007.

[16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, second edition, May 2002.

[17] S. E. Deering and D. R. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Trans. Comput. Syst.*, 8(2):85–110, 1990.

[18] DETER Network Security Testbed. `http://www.isi.deterlab.net`.

[19] P. Dewan, D. K. Naidu, and D. M. Durham. Denial of Service Attacks Using Internet Key Exchange Protocol. In *5th IEEE Consumer Communications and Networking Conference (CCNC'08)*, January 2008.

[20] DNP3 Users Group Technical Committee. Distributed Network Protocol version 3. `http://www.dnp.org/`.

[21] Electric Network Control Systems Standards Working Group of the Data Acquisition, Processing, and Control Systems Subcommittee of the IEEE Substations Committee. IEEE Standard Definition, Specification, and Analysis of Systems Used for Supervisory Control, Data Acquisition,and Automatic Control, March.

[22] S. Engineering Laboratories Inc. SEL-3021 Serial Encrypting Transceiver. `http://www.selinc.com/SEL-3021-1/`.

[23] S. Engineering Laboratories Inc. SEL-3022 Wireless Encrypting Transceiver. `http://www.selinc.com/SEL-3022/`.

[24] C. A. Gunter, S. Khanna, K. Tan, and S. Venkatesh. DoS Protection for Reliably Authenticated Broadcast. In *Proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS'04*, February 2004.

[25] C. A. Gunter, M. LeMay, J. Zhang, G. Gross, and S. T. King. Mitigating Risks and Exploiting Opportunities for Networked Power Devices. Technical report, March 2007.

[26] H. H. Hamed, E. S. Al-Shaer, and W. Marrero. Modeling and Verification of IPSec and VPN Security Policies. In *The 13th IEEE International Conference on Network Protocols (ICNP'05)*, November 2005.

[27] H. Harald Gjermundrød, D. E. Bakken, C. H. Hauser, and A. Bose. GridStat: A Flexible QoS-Managed Data Dissemination Framework for the Power Grid. *IEEE Transactions on Power Delivery*, 24(1):136 –143, January 2009.

[28] T. Hardjono and G. Tsudik. Ip Multicast Security: Issues and Directions. Technical report, Tech. Rep., Annales de Telecom, July-August, 1999.

[29] T. Hardjono and B. Weis. The Multicast Group Security Architecture. RFC 3740 (Informational), Mar. 2004.

[30] D. Harkins and D. Carrel. RFC2409, The Internet Key Exchange (IKE), November 1998.

[31] H. Harney, U. Meth, A. Colegrove, and G. Gross. GSAKMP: Group Secure Association Key Management Protocol. RFC 4535 (Proposed Standard), June 2006.

[32] C. H. Hauser, D. E. Bakken, and A. Bose. A Failure to Communicate: Next-Generation Communication Requirements, Technologies, and Architecture for the Electric Power Grid. *Power and Energy Magazine, IEEE*.

[33] A. Herzberg and H. Shulman. Stealth DoS Attacks on Secure Channels. In *Proceedings of the 17th Annual Network and Distributed System Security Symposium (NDSS'10*, March 2010.

[34] IEC TC 57/WG 10. IEC 61850-90-1: Use Of IEC 61850 For The Communication Between Substations, March 2010.

[35] IEC TC 57/WG 10-12. IEC61850 Communication Networks And Systems In Substations, April 2003.

[36] IEC TC 57/WG 15. IEC62351 Power Systems Management and Associated Information Exchange - Data and Communications Security, June 2007.

[37] IEC TC 57/WG 19. IEC 61850-90-2: Use of IEC 61850 For the Communication Between Control Centers And Substations (Draft), June 2008.

[38] IEEE-SA TR 1550. Utility Communications Architecture (UCA) Version 2.0. Technical report, November 1999.

[39] V. M. Igure, S. A. Laughtera, and R. D. Williamsa. Security Issues in SCADA Networks. *Computers & Security*, 25(7):498–506, October 2006.

[40] International Telecommunication Union, Telecommunication Standardization Sector (ITU-T). ITU-T Rec. X.200: Open Systems Interconnection - Basic Reference Model: The Basic Model, July 1994.

[41] A. Ipakchi and F. Albuyeh. Grid of the Future. *Power and Energy Magazine, IEEE*, 7(2):52 –62, March - April 2009.

[42] iproute2. `http://www.linuxfoundation.org/en/Net:Iproute2`.

[43] IPsec-Tools. `http://ipsec-tools.sourceforge.net/`.

[44] ISO Technical Committee 184 (TC184). ISO/IEC 9506: Industrial Automation systems - Manufacturing Message Specification, 2003.

[45] C. Kaufman. RFC4306, Internet Key Exchange (IKEv2) Protocol, December 2005.

[46] H. Khurana, R. Koleva, and J. Basney. Performance of Cryptographic Protocols for High-Performance, High-Bandwidth and High-Latency Grid Systems. In *E-SCIENCE '07: Proceedings of the Third IEEE International Conference on e-Science and Grid Computing*, pages 431–439, Washington, DC, USA, 2007. IEEE Computer Society.

[47] C. Kropiwiec, E. Jamhour, and C. Maziero. A Framework for Protecting Web Services with IPsec. In *Euromicro Conference*, pages 290 – 297, September 2004.

[48] M. LeMay, G. Gross, C. A. Gunter, and S. Garg. Unified Architecture for Large-Scale Attested Metering. In *Hawaii International Conference on System Sciences*, Big Island, Hawaii, January 2007. ACM.

[49] Q. Li and W. Trappe. Staggered TESLA: A Multicast Authentication Scheme Resistant to DoS Attacks. In *IEEE Global Telecommunications Conference (GLOBECOM'05)*, volume 3, November 2005.

[50] Y. Liang and R. H. Campbell. Understanding and Simulating the IEC 61850 Standard. Technical Report UIUCDCS-R-2008-2967, University of Illinois at Urbana-Champaign, May 2008.

[51] Y. Liu, M. K. Reiter, and P. Ning. False Data Injection Attacks Against State Estimation in Electric Power Grids. In *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security*, pages 21–32, New York, NY, USA, 2009. ACM.

[52] J. McCain and R. Straayer. Newnan Utilities SCADA Network Operates Over Encrypted Internet. *Electric Energy T & D*, 11(June):30 – 32, 2007.

[53] S. Mix. Secure substation communications, April 2007. Presentation on ITI Special Seminar, UIUC Granger Lecture Series,.

[54] Modbus Organization. Modbus Application Protocol Specification. `http://www.modbus.org`.

[55] V. Nikov. A DoS Attack Against the Integrity-Less ESP (IPSEC),.

[56] North American SynchroPhasor Initiative (NASPI). `http://www.naspi.org/`.

[57] J. Northcote-Green and R. G. Wilson. *Control and Automation of Electrical Power Distribution Systems*. Power Engineering. CRC Press, first edition, September 2006.

[58] S. Pal. Power Grid 2.0 - A Journey from Unified Communication to Unified Grid.

[59] A. Pannetrat and R. Molva. Efficient Multicast Packet Authentication. In *Proceedings of the 10th Annual Network and Distributed System Security Symposium*, 2003.

[60] J. M. Park, E. K. Chong, and H. J. Siegel. Efficient Multicast Packet Authentication Using Signature Amortization. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, 2002.

[61] A. Perrig, R. Canetti, J. D. Tygar, and D. Song. Efficient Authentication and Signing of Multicast Streams Over Lossy Channels. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pages 56–73, 2000.

[62] A. Risley, J. Roberts, and P. LaDow. Electronic Security of Real-time Protection And SCADA Communications. In *5th Annual Western Power Delivery Automation Conference*, Spokane, Washington, April 2003.

[63] S. Rowles, A. Yeung, and P. Tran. IETF Internet-Draft: Group Key Management using IKEv2, draft-yeung-g-ikev2-01, March 2010.

[64] J. L. Rrushi. *Composite Intrusion Detection in Process Control Networks*. PhD thesis, The University of Milan, January 2009.

[65] A. T. Sherman and D. A. McGrew. Key Establishment in Large Dynamic Groups Using One-Way Function Trees. *IEEE Transactions on Software Engineering*, 29(5):444–458, 2003.

[66] T. S. Sidhu, M. G. Kanabar, and P. P. Parikh. Implementation issues with iec 61850 based substation automation systems. In *Fifteenth National Power Systems Conference (NPSC'08)*, December 2008.

[67] T. Skeie, S. Johannessen, and C. Brunner. Ethernet in Substation Automation. *IEEE Control Systems Magazine*, 22(3):43 – 51, June 2002.

[68] B. P. Smith and D. R. Highfill. Tva investigates end-to-end integration. *Transmission and Distribution World*, pages 20 –26, July 2007.

[69] L. Standards Committee of the IEEE Computer Society. IEEE Std 802.1ae Media Access Control (MAC) Security, August 2006.

[70] K. Stouffer, J. Falco, and K. Scarfone. Guide to Industrial Control Systems (ics) Security. Technical report, National Institute of Standards and Technology (NIST).

[71] strongSwan. `http://www.strongswan.org/`.

[72] Substation Committee of the IEEE Power Engineering Society. IEEE Std 1646 Standard Communication Delivery Time Performance Requirements for Electric Power Substation Automation, 2005.

[73] J. Sun, W. Sheng, S. Wang, and K. Wu. Substation Automation High Speed Network Communication Platform based on MMS+TCP/IP+Ethernet. In *International Conference on Power System Technology*, volume 2, pages 1296–1300, Octorber 2002.

[74] The XML C parser and toolkit of Gnome. `http://xmlsoft.org/`.

[75] J. D. Touch and Y.-H. E. Yang. Reducing the Impact of DoS Attacks on Endpoint IP Security. In *2nd IEEE Workshop on Secure Network Protocols (NPSEC06).*, pages 6 –11, November 2006.

[76] S. Tuttle, G. Pizano, and C. Smith. *AIX 5L Version 5.2 Security Supplement*. IBM Redbooks, November 2003.

[77] U.S. - Canada Power System Outage Task Force. Final Report on the August 14, 2003 Blackout in the United States and Canada: Causes and Recommendations, April 2004. `https://reports.energy.gov/B-F-Web-Part1.pdf`.

[78] U.S. Government Accountability Office. Critical Infrastructure Protection: Multiple Efforts to Secure Control Systems Are Under Way, but Challenges Remain. Technical report, September 2007.

[79] Q. Wang, H. Khurana, Y. Huang, and K. Nahrstedt. Time Valid One-Time Signature for Time-Critical Multicast Data Authentication. In *IEEE 2009 Conference on Computer Communications (INFOCOM'09)*, 2009.

[80] B. Weis, G. Gross, and D. Ignjatic. Multicast Extensions to the Security Architecture for the Internet Protocol. RFC 5374 (Proposed Standard), Nov. 2008.

[81] C. K. Wong, M. Gouda, and S. S. Lam. Secure Group Communications Using Key Graphs. *IEEE/ACM Transactions on Networking*, 8(1):16 –30, February 2000.

[82] C. K. Wong and S. S. Lam. Digital Signatures for Flows and Multicasts. *IEEE/ACM Trans. Netw.*, 7(4):502–513, 1999.

[83] A. Wool. A Quantitative Study of Firewall Configuration Errors. *IEEE Computer*, 37(6):62 – 67, 2004.

[84] F. F. Wu, K. Mosleshi, and A. Bose. Power System Control Centers: Past, Present,and Future. In *Proceedings of The IEEE*, volume 93, pages 1890 – 1908, November 2005.

[85] H. Ying and H. Wang. Building An Application-aware IPsec Policy System. *IEEE/ACM Trans. Netw.*, 15(6):1502–1513, 2007.