Application-aware secure multicast for power grid communications

Jianqing Zhang* and Carl A. Gunter

Department of Computer Science,
University of Illinois at Urbana-Champaign,
Urbana, IL 61801, USA
E-mail: jzhang24@cs.illinois.edu
E-mail: jianqing.zhang@intel.com
E-mail: cgunter@cs.illinois.edu
*Corresponding author

Abstract: We propose an application-aware approach to setting up secure multicast groups for power grid communications that automatically derives group memberships and verifies configuration conformance from data dependencies in system specifications. We design an abstract multicast model, analysis algorithms, and configuration derivation techniques. These are implemented in a prototype system, SecureSCL. We also provide experimental evidence that IPsec multicast can address latency constraints in power substation networks.

Keywords: power grid communications; multicast; network security; application-aware; SecureSCL.

Reference to this paper should be made as follows: Zhang, J. and Gunter, C.A. (2011) 'Application-aware secure multicast for power grid communications', *Int. J. Security and Networks*, Vol. 6, No. 1, pp.40–52.

Biographical notes: Jianqing Zhang received his BS from the Beijing University of Aeronautics and Astronautics in 1998, his MS from the University of Pennsylvania in 2004 and his PhD from the University of Illinois at Urbana-Champaign in 2010. His research areas are security, networking and power grid communication. He is currently a research scientist at Intel Labs.

Carl A. Gunter received his BA from the University of Chicago in 1979 and his PhD from the University of Wisconsin at Madison in 1985. He worked as a postdoctoral researcher at Carnegie-Mellon University and the University of Cambridge in England before joining the Faculty of the University of Pennsylvania in 1987 and the University of Illinois in 2004. He is a Professor in the Computer Science Department, director of Illinois Security Lab, and a member of the Information Trust Institute (ITI). He has made research contributions in the semantics of programming languages, formal analysis of networks and security, and privacy.

1 Introduction

Multicast plays an important role in electric power grid systems. For example, IP multicast is considered in Phasor Measurement Units (PMUs) for delivering status data periodically in a large geographic area. UDP multicast is used in DNP3 to reset counters or values of multiple remote control devices near simultaneously. In IEC 61850 (IEC TC 57/WG 10-12, 2003) 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.

As power grid communications are migrating from industry proprietary infrastructures to public infrastructures and protocols (Wu et al., 2005; Ping Sun et al., 2002; Skeie et al., 2002), cyber security risks are also increased beyond those encountered when such systems rely on physical isolation for protection. It is expected that most of the vulnerabilities existing in the internet could also occur in power grid networks (Powner and Rhodes, 2007; Igure et al., 2006; Liu et al., 2009). Security, especially integrity, of multicast will be one of the most interesting and challenging problems for power grid systems.

Secure multicast solutions for power grid must address some particular challenges when providing security guarantees with cryptographically secured protocols. First, because of intricate system designs, the need to integrate proprietary configuration tools from different vendors, and the complexity of configuring off-the-shelf security protocols, it is a complex and error-prone task to configure group memberships, policies and group keys. Besides functional mistakes, misconfiguration may lead to security violations. For example, sensitive data could be delivered to a wrong device due to incorrect group partitions. An automatic and error-resistant configuration mechanism would improve the efficiency and mitigate inconsistency or mistakes in system design and deployment. Second, various application requirements (Substation Committee of the IEEE Power Engineering Society, 2005) 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 within 2 ms and 10 ms. PMU has frequency requirements at 30 times per second or even higher. Naive approaches to securing these messages usually do not succeed in meeting the latency requirements. For example, IEC 62351-6 (IEC TC 57/WG 15, 2007) relies on public key signatures on each GOOSE frame and is not able to guarantee timing requirements because of the latency impact of such signatures. Third, a feasible and integrated group key management scheme is required for secure power grid multicast systems. Although researchers already proposed a number of sophisticated group key management protocols or schemes, most of them are complicated and hard to integrate with power grid systems smoothly. The configuration of the group key management protocols is also a big challenge. Fourth, the balance amongst efficiency, feasibility and cost must be considered carefully. It would be 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.

In this paper, we propose an application-aware approach to setting up multicast groups for power grid communications using network layer security. The basic idea is to derive group memberships and publish–subscribe relationships based on data dependencies, which are derived from an appropriate extension of system domain-specific specifications. This is based on the observation that the data is the focus of a publish–subscribe system and associates all group members in a multicast application. We derive multicast groups by integrating the network layer group management with the application layer functional configurations. On the basis of the derived group memberships, we try to detect inconsistent configurations using a configuration verification tool. It will help power engineers correct system configuration mistakes and facilitate the system design. Furthermore, the approach configures the group key exchange protocol based on application logic and integrates the group key management system with the multicast system seamlessly. We also 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 capability of wide area multicast for inter-substation communications and the support from commercial IPsec implementations.

We design a multicast model and a publish–subscribe model to formally present multicast systems and publish–subscribe relationships. We classify a number of multicast configuration anomalies and develop the algorithms to detect the anomalies. Both the models and the algorithms are the enhancement of the work in Zhang and Gunter (2010). They are more generic and more extensible for future work. A multicast and group key management architecture based on the Group Domain of Interpretation (GDOI) (Baugher et al., 2003) is designed 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 the network layer secure multicast configuration to the application-specific configuration of power substations.

To demonstrate this methodology, we take IEC 61850 power substation networks as a case study and develop a prototype system *SecureSCL*. SecureSCL extracts multicast groups for GOOSE from extended Substation Configuration Language (SCL). It 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 on a portion of the SCL specification of a power substation of the Tennessee Valley Authority (TVA).

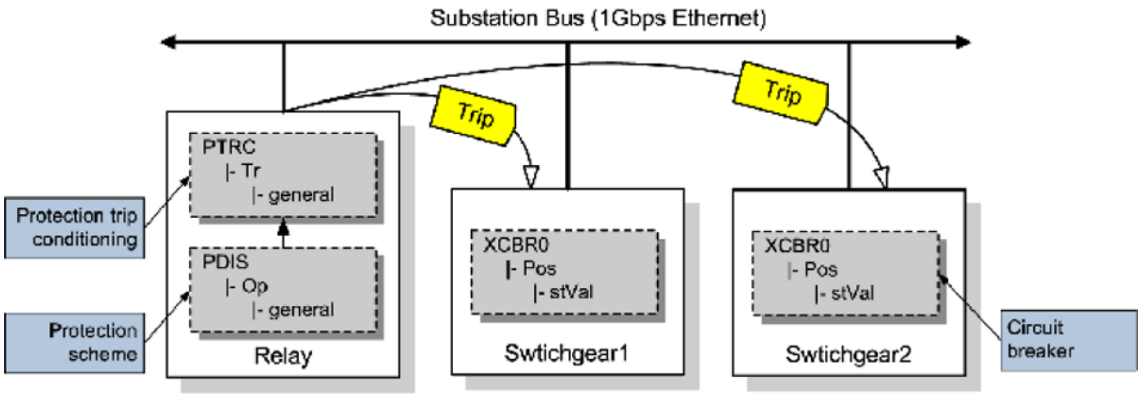
The rest of the paper is organised as follows. Section 2 reviews background on power substation communication and related works. A formal model depicting multicast applications in substation networks is presented in Section 3. Section 4 discusses the multicast configuration anomalies and the detection algorithms. We show the implementation of the system and the case study in Section 5. The performance of IPsec-based multicast is discussed in Section 6. Section 7 provides the case study of secureSCL Section 7 concludes and discusses the future work.

2 Background and related work

2.1 IEC 61850

A power substation is a subsidiary station of an electricity generation, transmission and distribution system. A power substation network usually consists of tens or hundreds of microprocessor-based IEDs that control, monitor, and protect the power grid.

Figure 1 Timing-critical multicast in power substation (see online version for colours)



Nowadays, IEDs are increasingly connected by ethernet for transmitting status data, control commands and configuration information.

IEC 61850 (IEC TC 57/WG 10-12, 2003) is a specification for the design and configuration of substation automation. It supports a comprehensive set of substation functions and provides rich features for substation communications. It is also extensible enough to support system evolution. IEC 61850 uses object-oriented data models to describe the information of various primary equipments and substation automation functions. It specifies the communication interfaces between IEDs and the schemes mapping them to a number of protocols running over TCP/IP and high-speed ethernet.

GOOSE is a link-layer multicast protocol designed in IEC 61850 for transmitting timing-critical messages, such as substation events, commands and alarms, within power substation networks. Because GOOSE is directly mapped to ethernet frames, it can take advantage of high-speed switched ethernet and is capable of fulfilling timing requirements. Figure 1 illustrates an example where GOOSE is used to prevent a fault from being propagated. A protective relay multicasts a TRIP command to two circuit breakers to disconnect the circuits upon detecting a fault. The logical node PDIS represents a protection scheme by the data object Op, and the logical node PTRC represents a protection trip conditioning by the data object Tr. Usually, the values of these two data objects, i.e., the TRIP, must be transmitted within a few milliseconds. It is common to quote a benchmark of 4 ms for this threshold so we use that timing constraint in this paper. The 4 ms threshold is easily and reliably met by ethernet multicast on commodity hardware at the load levels seen in power substations.

IEC 61850 defines the XML-based SCL for inter-operable exchange of system configuration data between different vendors and configuration tools. It is a global description of a substation 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. SCL defines an object model describing 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. Figure 2 shows the partial SCL expression of the TRIP command in Figure 1. The DateSet element consists of a number of FCDA (functionally constrained data attribute) elements. Each FCDA is a reference to a data attribute of the data objects Tr and Op in the logical node PTRC1 and PDIS1.

Figure 2 TRIP Command in Substation Configuration Language

```
<DataSet name="dsTripLogic">
    <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>
```

2.2 Related work

Most existing secure multicast solutions from both academia and industry have some drawbacks for power grid systems. IEC 62351-6 (IEC TC 57/WG 15, 2007), for example, authenticates GOOSE frames using RSA signatures without assuring low-latency operation. Its short signature field also limits the authentication with large keys. It extends SCL to support certificates and secure access points, but no details are presented. IEEE802.1AE (LAN/MAN Standards Committee of the IEEE Computer Society, 2006) provides security for ethernet frames using a hub-and-spokes topology. Security associations are set up between a switch and each host. All frames between two host must be relayed by the switch, thus extra latency is introduced. Researchers have suggested a number of schemes (Wong and Lam, 1999; Perrig et al., 2000; Pannetrat and Molva, 2003; Park et al., 2002; Wang et al., 2009) for secure multimedia streams, which achieve the goals of integrity, fast-rate signature/verification and loss-tolerance. However, few of these complicated solutions are standardised or commercialised. It is hard for industry to deploy them in real facilities. Those standardised group key specifications like GDOI (Baugher et al., 2003) usually lack the application-level support and group management. Canetti et al. (2000) proposes an IPsec-based host architecture for multicast. They introduce the concept of Multicast Internet Key Exchange (MIKE), describe the functionalities of the architecture components and implement a prototype system for validation.

In contrast to the above-mentioned works, this paper is focused on multicast configuration and group management using a formal model based on application data dependency. The proposed multicast model and verification mechanism can be extended for generic secure communication configurations. It is also an original work of studying the feasibility of IPsec-based multicast in power grid systems.

3 Multicast modelling

3.1 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 1.

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 $m_2 = 3$.

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 monitoring 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. In this example, the switchgears only update the value of Pos_i. Figure 3 shows the illustrative diagram of the motivating example. The arrows show that the multicast data flows 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. For simplicity, we do not use the sophisticated naming conventions defined in IEC 61850 and simplify the data model. The transmission scheme used in GOOSE is also simplified.

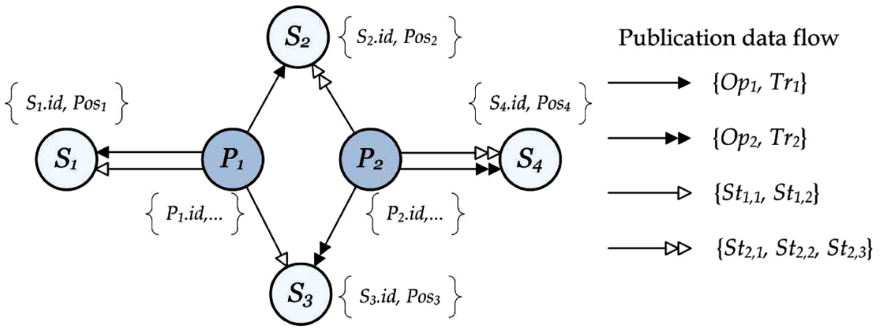
3.2 Elements of a secure multicast system

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 Pb, a set of subscribers S and a set of group controllers G. Components that have relationships with data objects are called *entities*, E. Therefore, O, C, P and S are entities, i.e., O, C, P, S $\subseteq$ E.

Data object is the core of the secure multicast model. In this work, our attention is focused on the *protection system data*, including physical parameters, environment conditions or control commands. They can be represented as a set of data objects and delivered using timing-critical multicast. 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.

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. A *data owner* is an entity that owns or hosts a number of data objects. Any control device with accessible data objects could be a data owner. For example, the protective relay P_1 is the owner of the data object Op_1. A *data consumer* is an entity whose operations need

Figure 3 Motivating example: multicast in GOOSE applications (see online version for colours)



certain data objects. For example, the switchgear S_1 needs the data objects sets $\{Op_1.Tr_1\}$ and $\{St_{1,1}, St_{1,2}\}$.

A *publisher* is a content provider and publishes data objects using multicast. It could be a protective relay that issues control commands, or a sensor that provides power status data to other devices. 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. However, a data owner may not be a publisher in the multicast model. For example, S_1 owns data object Pos_1 but it does not publish Pos_1. A *subscriber* is an entity that subscribes to data objects from publishers. It could be a circuit breaker that executes commands issued by relays, or a protective relay that monitors power grid via the status update from sensors. 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. A data consumer may require a variety of data objects and not all of them are delivered via multicast. That is, the data consumer may not be a subscriber in the model. 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 sensors, it behaves as a subscriber. Without loss of generality, we study the multicast groups individually and the roles of publishers and subscribers in the group do not change.

A *group controller* provides group membership and group key management service. A group controller performs the following tasks:

1 authorise group access privileges based on group memberships derived from system configurations

2 generate, distribute and refresh shared group keys

3 revoke group memberships based on changing configurations.

3.3 Publisher-subscriber model

The publisher–subscriber model describes the relations between entities and data objects in a multicast system and specifies a collection of functions based on the relations.

Ownership. $\mathcal{R}_{\text{own}} \subseteq \mathsf{E} \times \mathsf{D}$. If an entity e *owns* or *hosts* a data object d, then $(e, d) \in \mathcal{R}_{\text{own}}$ or $\mathcal{R}_{\text{own}}(e, d)$, where $e \in \mathsf{E}$ and $d \in \mathsf{D}$. Apparently, e is a data owner. For example, $\mathcal{R}_{\text{own}}(P_1.id, Op_1)$ and $\mathcal{R}_{\text{own}}(P_2.id, Tr_2)$. We define the function $\overline{\mathcal{R}_{\text{own}}} : \mathsf{E} \rightarrow 2^{\mathsf{D}}$ by the equation:

$$\overline{\mathcal{R}_{\text{own}}}(e) = \{d \in \mathsf{D} : (e, d) \in \mathcal{R}_{\text{own}}\}, \quad \text{where } e \in \mathsf{E}.$$

It returns the set of data objects that are owned by an entity e. For example, $\overline{\mathcal{R}_{\text{own}}}(S_3) = \{S_3.id, Pos_3\}$.

Publication. $\mathcal{R}_{\text{pub}} \subseteq \mathsf{E} \times 2^{\mathsf{D}}$. If an entity e *publishes* a data set ds, then $(e, ds) \in \mathcal{R}_{\text{pub}}$ or $\mathcal{R}_{\text{pub}}(e, ds)$, where $e \in \mathsf{E}$ and $ds \in 2^{\mathsf{D}}$. Apparently, e is a publisher. In the running example, $\mathcal{R}_{\text{pub}}(P_1, \{Op_1, Tr_1\})$ and $\mathcal{R}_{\text{pub}}(P_2, \{St_{2,1}, St_{2,2}, St_{2,3}\})$. Note that $\mathcal{R}_{\text{own}}$ and $\mathcal{R}_{\text{pub}}$ are two independent relations specified 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 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 the function $\widehat{\mathcal{R}_{\text{pub}}} : \mathsf{P} \rightarrow 2^{2^{\mathsf{D}}}$ by the equation:

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

It returns the union set of data sets which are published by a publisher p. For example, $\widehat{\mathcal{R}_{\text{pub}}}(P_2) = \{\{Op_2, Tr_2\}, \{St_{2,1}, St_{2,2}, St_{2,3}\}\}$. Actually, for a given publisher $p \in \mathsf{P}$, the number of members of $\widehat{\mathcal{R}_{\text{pub}}}(p)$ 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. Further, we define a function $\overline{\mathcal{R}_{\text{pub}}} : \mathsf{P} \rightarrow 2^{\mathsf{D}}$ by the equation:

$$\overline{\mathcal{R}_{\text{pub}}}(p) = \bigcup \widehat{\mathcal{R}_{\text{pub}}}(p), \quad \text{where } p \in \mathsf{P}.$$

It returns the set of data objects that are published by a publisher p. For example, $\overline{\mathcal{R}_{\text{pub}}}(P_2) = \{Op_2, Tr_2, St_{2,1}, St_{2,2}, St_{2,3}\}$. Given a publisher p, we can check if it has illegitimate publications by comparing the result of $\overline{\mathcal{R}_{\text{pub}}}(p)$ and $\overline{\mathcal{R}_{\text{own}}}(p)$. If $\overline{\mathcal{R}_{\text{pub}}}(p)$ is not a subset of $\overline{\mathcal{R}_{\text{own}}}(p)$, p publishes one or more data objects that do not belong to it.

Consumption. $\mathcal{R}_{\text{con}} \subseteq \mathsf{E} \times 2^{\mathsf{D}}$. If an entity $e \in \mathsf{E}$ *requires* a data object set, $ds \in 2^{\mathsf{D}}$, then $(e, ds) \in \mathcal{R}_{\text{con}}$ or $\mathcal{R}_{\text{con}}(e, ds)$. For example, $\mathcal{R}_{\text{con}}(S_1, \{Op_1, Tr_1\})$ and $\mathcal{R}_{\text{con}}(S_3, \{St_{1,1}, St_{1,2}\})$. Apparently, e is a data consumer. 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.

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 $\widehat{\mathcal{R}_{\text{con}}} : \mathsf{C} \rightarrow 2^{2^{\mathsf{D}}}$ by the equation:

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

It returns the union of data sets that are consumed by c. For example, $\widehat{\mathcal{R}_{\text{con}}}(S_3) = \{\{Op_2, Tr_2\}, \{St_{1,1}, St_{1,2}\}\}$. We also define the function $\overline{\mathcal{R}_{\text{con}}} : \mathsf{C} \rightarrow 2^{\mathsf{D}}$ by the equation:

$$\overline{\mathcal{R}_{\text{con}}}(c) = \bigcup \widehat{\mathcal{R}_{\text{con}}}(c), \quad \text{where } c \in \mathsf{C}.$$

It returns the set of data objects that are consumed by c. For example, $\overline{\mathcal{R}_{\text{con}}}(S_3) = \{Op_2, Tr_2, St_{1,1}, St_{1,2}\}$.

Subscription. $\mathcal{R}_{\text{sub}} \subseteq \mathsf{E} \times \mathsf{E} \times 2^{\mathsf{D}}$. If s *subscribes* to ds from p, then $(s, p, ds) \in \mathcal{R}_{\text{sub}}$ or $\mathcal{R}_{\text{sub}}(s, p, ds)$. For a given $(s, p, ds) \in \mathcal{R}_{\text{sub}}$, the relation represents a subscription request sent by $s \in \mathsf{E}$, i.e., s is a subscriber.

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 authorised finally. First of all, s only can subscribe the data object that it has access to, i.e., the subscribed data set ds must be a subset of one of the 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 the data sets p publishes, i.e., p does publish a data set that contains all data objects in ds. If all the above-mentioned requirements are satisfied, we call such a subscription *valid subscription*. Formally, a subscription $(s, p, ds) \in \mathcal{R}_{\text{sub}}$ is *valid* if, and only if

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

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 Multicast configuration anomaly

In this section, we discuss four types of anomalies in multicast configuration, especially those occurring in IEC 61850 multicast applications. We represent the formal description of each anomaly and design algorithms to detect these anomalies based on the multicast model in Section 3.

4.1 Anomaly classification

Ownership anomaly. A publisher p publishes a data set ds, which consists of data objects that are not owned by p. Formally, a publication $\mathcal{R}_{\text{pub}}(p, ds)$ has an *ownership anomaly* if:

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

Also, a publisher p has a *publication ownership anomaly* if:

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

For example, in the motivating example, if a publication $\mathcal{R}_{\text{pub}}(P_1, \{Op_1, Tr_2\})$ 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, it will have a publication ownership anomaly. Such anomaly usually occurs when the system data flow design is changed but the IED's publication configuration does not change accordingly,

Publication redundancy. A publisher p publishes a data set ds, but no entity consumes or subscribes to it. 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. Formally, a publication $\mathcal{R}_{\text{pub}}(p, ds)$ is *fully redundant* if:

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

In the partial redundancy, partial data objects in the data set are requested by some data consumers, while others are not, i.e., a publication $\mathcal{R}_{\text{pub}}(p, ds)$ is *partially redundant* if:

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

In the motivating example, a publication of $\mathcal{R}_{\text{pub}}(P_2, \{P_2.id\})$ would be a full redundant publication since no switchgear requests the relay's id. A publication of $\mathcal{R}_{\text{pub}}(P_2, \{P_2.id, Op_2, Tr_2\})$ 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. Partial redundancy happens when the subscribers only need parts of the data object set. Since a publication may be subscribed by the data consumers, which have different requirements, 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.

Source anomaly. A subscriber s requests a subscription to a data set ds published by a publisher p, but p does not exist in the system. Formally, a subscription request $\mathcal{R}_{\text{sub}}(s, p, ds)$ has a *source anomaly* if $p \notin \mathsf{E}$.

In the motivating example, a subscription request of $\mathcal{R}_{\text{sub}}(S_1, P_3, \{Op_1, Tr_1\})$ 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.

Data dissatisfaction. Given a subscription request $\mathcal{R}_{\text{sub}}(s, p, ds)$, no publication (p, ds') can provide all data objects requested in the subscription. 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, a subscription request $\mathcal{R}_{\text{sub}}(s, p, ds)$ is *hard data dissatisfactory* if:

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

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, i.e., a subscription request $\mathcal{R}_{\text{sub}}(s, p, ds)$ is *soft data dissatisfactory* if:

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

For example, a subscription request of $\mathcal{R}_{\text{sub}}(S_1, P_1, \{Op_1, Tr_1, P_1.id\})$ would be hard data dis-satisfactory since P_1 does not publish $P_1.id$. A subscription request of $\mathcal{R}_{\text{sub}}(S_1, P_1, \{Op_1, Tr_1, St_{1,2}\})$ would be 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.

4.2 Anomaly detection algorithms

Algorithm 1 detects ownership anomaly. It takes a publisher p as an input and returns a data object set namely DSet, which contains the data objects that are not owned by p but published by it. If DSet is empty, the publisher p has no publication ownership anomaly. 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 $\overline{\mathcal{R}_{\text{own}}}(p)$ and put them into DKeys. After sorting the list using the quicksort algorithm (Cormen et al., 2002) 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.

Algorithm 2 detects both full publication redundancy and partial publication redundancy. The algorithm takes a publication $\mathcal{R}_{\text{pub}}(p, ds)$ 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 usage of DKeys is same as Algorithm 1. It 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 $\mathcal{R}_{\text{pub}}(p, ds)$ and searches the

Algorithm 1: Detect Ownership Anomaly

```
   input : $p$
   output: DSet
 1 begin
 2     DKeys ← $nil$;
 3     DSet ← $\emptyset$;
 4     for $d \in \overline{\mathcal{R}_{own}}(p)$ do
 5         key ← hash ($d$) ;
 6         appendKey (DKeys, key) ;
 7     end
 8     quickSort (DKeys) ;
 9     for $d' \in \overline{\mathcal{R}_{pub}}(p)$ do
10         key ← hash ($d'$) ;
11         result ← binarySearch (DKeys, key) ;
12         if result = $nil$ then
13             appendSet (DSet, $d'$) ;
14         end
15     end
16 end
```

Algorithm 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
```

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 (Lines 20–26).

Algorithm 3 detects source anomaly. It takes a subscription request $\mathcal{R}_{\text{sub}}(s, p, ds)$ and the whole entity set E as the inputs, and returns the status of the subscription: source-anomaly or clear. The algorithm calculates and stores the hash values of all entities (keys) in a list EKeys (Lines 3–6). After sorting the list by keys (Line 7), the algorithm searches the list for the key of the publisher p (Lines 8 and 9). If the search fails, this subscription has a source anomaly, otherwise its status is cleared (Lines 10–14).

Algorithm 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
```

We design Algorithm 4 to detect both hard-dissatisfaction and soft-dissatisfaction anomalies. It takes a subscription $\mathcal{R}_{\text{sub}}(s, p, ds)$ 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.

Algorithm 4: Detect Data Dissatisfaction

```
   input  : $\mathcal{R}_{sub}(s,p,ds)$
   output: status
1  begin
2      PubSet ← $\emptyset$;
3      pubsetn ← 0 ;
4      subsetn ← countSet ($ds$) ;
5      satisfied ← true ;
6      for $ds_p \in \widehat{\mathcal{R}_{pub}}(p)$ do
7          quickSortSet ($ds_p$) ;
8          for $d \in ds$ do
9              result ← binarySearchSet ($ds_p$, $d$) ;
10             if result = $nil$ then
11                 satisfied ← false ;
12             else
13                 appendSet (PubSet, $d$) ;
14             end
15         end
16         if satisfied = true then
17             status ← clear ;
18             break ;
19         else
20             satisfied ← true ;
21         end
22     end
23     pubsetn ← countSet (PubSet) ;
24     if status $\neq$ clear then
25         if pubsetn < subsetn then
26             status ← hard-dissatisfaction ;
27         else if pubsetn = subsetn then
28             status ← soft-dissatisfaction ;
29         end
30     end
31 end
```

The variable PubSet 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 $\mathcal{R}_{\text{sub}}(s, p, ds)$ 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 Lines 6–22. It checks all data object sets published by p to see if there is a publication that 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 $\mathcal{R}_{\mathrm{sub}}(s, p, ds)$ can be satisfied by current publication $\mathcal{R}_{\mathrm{pub}}(p, ds_p)$. Its status will be set as **clear** and the outside loop can break (Lines 16–21). 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 (Lines 23–30). 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.

5 Implementation and case study

5.1 Overview

Figure 4 shows the host architecture of SecureSCL. We partitioin the system into three planes: *configuration plane*, *control plane* and *data plane*. The system works in three phases: *design phase*, *initialisation phase* and *running phase* (see Figure 5).

The configuration plane works in the design phase. A configuration language parser parses system specification and configuration files. On the basis of the parser's output, a multicast model and consistency analyser sets up the data model and the publish–subscribe model presented in Section 3. It also checks configuration correctness and consistency using the anomaly detection algorithms. If an anomaly is detected, the system will go back to the step of configuration revision. Otherwise, the system enters the initialisation phase and the control plane takes over.

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. 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 and management flow*. 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 composed of upper layer applications, like GOOSE

Figure 4 Host architecture (see online version for colours)

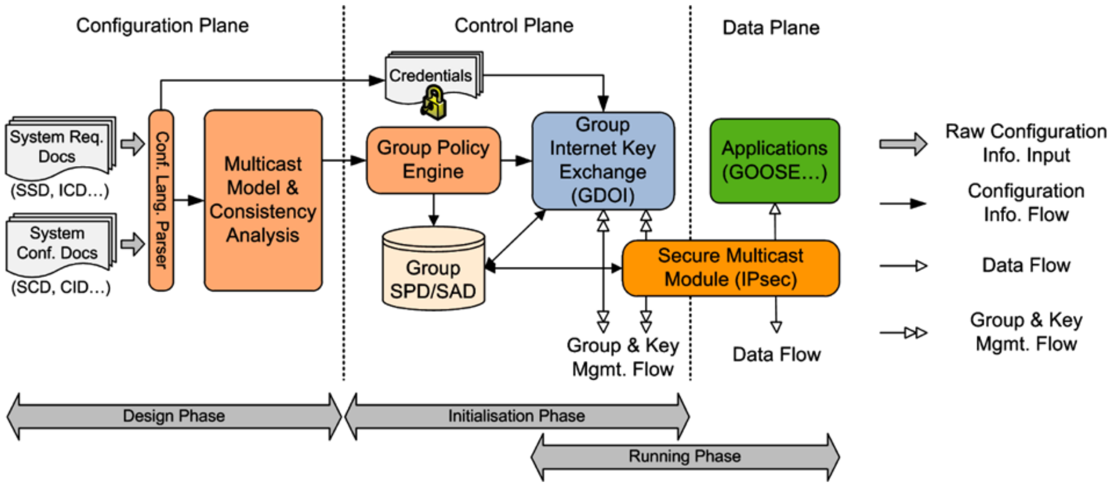
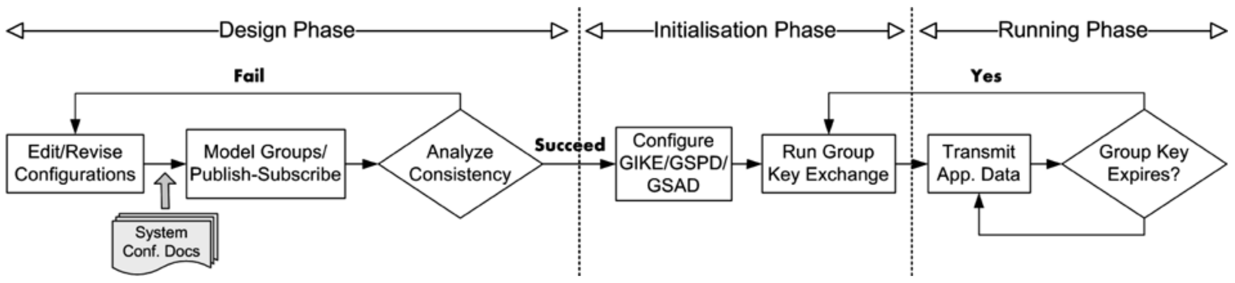


Figure 5 System working phases



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*. At the same time, the GIKE module also makes use of the SMM to securely refresh group session keys periodically without interfering the data flow. So, both the control plane and the data plane work during the running phase.

The whole system is implemented using C/C++ on Ubuntu 8.04. The extended configuration language parser is developed using libxml (http://xmlsoft.org/). The GPE module makes use of NETLINK sockets to manipulate IPsec SPD and SAD. A reference implementation of GDOI from Cisco is used for the GIKE module.

5.2 Design

Security Extended Configuration. To support secure multicast, especially IPsec-based multicast, a configuration language needs to be extended for more information, including:

- credentials (or their references) required for group key exchange
- additional network 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. In this work, we base the implementation on SCL and integrate security credentials using XMLDSIG (Bartel et al., 2008). We show the extension to the network configuration to raise GOOSE to the network layer, the integration of security information, especially the credentials used for group key exchange, and the specification of the group controller.

Group Policy Engine. The GPE transforms the multicast model to group authorisation policy and traffic policy. Authorisation 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 when it is processing multicast packets. The GPE also transforms these policies to a configuration file recognisable to the GIKE (GDOI).

The GPE module on a group member provides the information about the multicast group, the group controller and the configuration profiles for running group key exchange protocols like security credentials. It also invokes group key negotiation with the group controller. For a group controller, GPE directs GIKE for group authorisation. on the basis of the group associations derived from the multicast model, GPE provides GIKE with the group information like the multicast groups addresses, the identities of the authorised group members and their security credentials, and parameters for group management like the interval of refreshing group keys, etc. It also invokes the GIKE module listening to group key negotiation requests.

Group Internet Key Exchange. The GIKE module is a protocol used for group membership authorisation and group key management. In this paper, we have borrowed the idea of a multicast group key management architecture from Hardjono and Weis (2004) and Baugher et al. (2005), and take advantage of the GDOI (Baugher et al., 2003), a centralised multicast security and key management protocol. 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.

Secure Multicast Module. We have based our design on the IPsec protocol suite. The IPsec protocol suite is a mature and sophisticated solution for secure data communication and key management and used widely. It has undergone a degree of formal analysis demonstrating that it preserves a variety of security properties. IPsec implementations on most off-the-shelf operating systems are able to protect multicast packets natively (Aurisch and Karg, 2003; Canetti et al., 2000). 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 (Zhang, 2010). Such mechanism avoids the packet replication and the additional latency due to multiple hops that occurs in the hub-and-spokes schemes like Casado et al. (2007) and LAN/MAN Standards Committee of the IEEE Computer Society (2006), and ensures all recipients can receive the message near simultaneously. Compared with some link layer security solutions, like IEC 62351 and IEEE 802.1AE, IPsec-based multicast is able to support critical multicast applications across wide area networks like PMU applications and GOOSE messages between substations or between substations and control centres (IEC TC 57/WG 10, 2010). Furthermore, our experiments show that IPsec multicast is adequately scalable and able to maintain latencies well below the 4ms target for substations of increasing sizes (see Section 6). Since IEC 61850 enabled IEDs usually have strong computing and networking capabilities, it is not very challenging for this class of control systems to utilise sophisticated security technologies like IPsec.

5.3 Case study

On the basis a portion of TVA Bradley IEC 61850 substation configuration (Smith and Highfill, 2007), we develop a case study to demonstrate the usability of SecureSCL. SecureSCL extends SCL by integrating new elements representing IPsec multicast, principals' credentials, etc. The element KeyInfo defined in XML Signature is used to describe security credentials. The extended configuration language parser is developed using libxml. The GPE module makes use of NETLINK sockets to manipulate IPsec SPD and SAD. A reference implementation of GDOI from Cisco is used for GIKE.

We integrate security configurations into SecureSCL. The element AccessPoint, which is used to specify an IED's communication interfaces, is extended by inserting IED's credentials using ds:KeyInfo. The Communication element describes the substation network topology including all IEDs' access points. GCKS is added to specify the group controller and the protocol. These information will be used for the GPE to configure the group key management protocol. A revised sscl:GSE element is used to assign class-D IP addresses for GOOSE, rather than the link layer interface (please see the details of the case study in Zhang (2010)).

6 Performance of IPsec-based multicast

We design a Process Control Emulation System (PCES) for emulating IPsec-protected GOOSE-like multicast within an ethernet LAN, and measuring round trip latencies. When a multicast request is sent by a publisher, PCES calculates the latency from a randomly chosen recipient. We argue that the sampled round trip latency measurement method can collect precise data. Given that all hosts have same computation capacity and connected with same bandwidth links, we assume all recipients receive the request and respond simultaneously. The duration, from the time when the publisher sends the request to the time when all subscribers receive the request and get ready to respond, is just the *application-to-application communication time* defined in Substation Committee of the IEEE Power Engineering Society (2005). The test is repeated 1000 times per round and the latencies are measured from different recipients.

PCES is written in C/C++ and deployed on the DETER Testbed (http://www.isi.deterlab.net), a public facility for medium-scale experiments in computer security. Our testbed consists of PCs running Ubuntu 8.04 with Linux kernel version 2.6.24 and uses strongSwan (http://www.strongswan.org/) for IPsec configuration. Both HMAC-SHA1 and AES are used for IPsec ESP.

We run the experiments for the network sizes of 4, 8, 16, 32 and 64 hosts, respectively. We randomly pick one publisher and have others listen and acknowledge. The publisher multicasts requests (140-byte UDP payload) 1000 times and subscribers respond with same size acknowledgements.

Figure 6 shows the results for 4/8/16/32-host scenarios in a 1Gbps switched ethernet LAN and 8/64-host scenarios in a 100 Mbps LAN with 16 and 32 hosts, respectively. The plot shows: when the network size increases from 8 hosts to 32 hosts, most latencies 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 20 and 25us (see Table 1, Note the scenarios of 8* and 16* are tested in a 100 Mbps LAN).

Figure 6 Latencies of native IPsec multicast

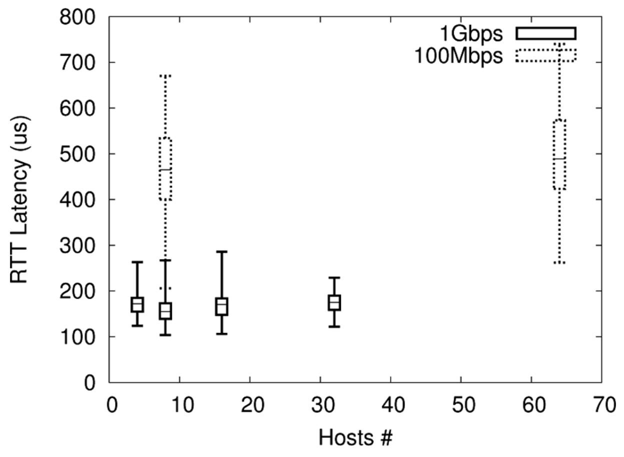


Table 1 Average and standard deviations of round trip latency

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

Although both the average latencies and the standard deviations in 100Mbps LANs are much larger, the data shows that native IPsec multicast is capable of fast packets transmission even the network bandwidth is limited. As the network size increases, its performance is not degraded remarkably. In general, native IPsec multicast is quite scalable and able to maintain latencies well below the 4ms target for substation networks of increasing sizes.

7 Conclusion

The application-aware secure multicast architecture is an efficient solution for multicast applications in power grid systems. By analysing derived multicast models and checking data dependencies based on functional configurations, it automates group management and minimises 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 4 ms 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.

Acknowledgements

We are grateful for help and encouragement we received from Rakesh Bobba, Chris Grier, Dong Jin, Himanshu Khurana, Samuel King, Michael LeMay, Bruce Muschlitz and Tim Yardley. This work was supported in part by DOE DE-OE0000097, HHS 90TR0003-01, NSF CNS 09-64392, NSF CNS 09-17218, NSF CNS 07-16626, NSF CNS 07-16421, NSF CNS 05-24695, and grants from the Department of Energy, the MacArthur Foundation, Boeing Corporation, and Lockheed Martin Corporation. The views expressed are those of the authors only.

References

Aurisch, T. and Karg, C. (2003) 'Using the IPsec architecture for secure multicast communications', *8th International Command and Control Research and Technology Symposium (ICCRTS)*, Washington DC.

Bartel, M., Boyer, J., Fox, B., Lamacchia, B. and Simon, E. (2008) 'XML-signature syntax and processing', in Eastlake, D., Reagle, J. and Solo, D. (Eds.): *W3C Recommendation*, June.

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

Baugher, M., Weis, B., Hardjono, T. and Harney, H. (2003) *The Group Domain of Interpretation*, RFC 3547, IETF, July.

Canetti, R., Chen Cheng, P., Giraud, F., Pendarakis, D., Rao, J.R., Rohatgi, P. and Saha, D. (2000) 'An IPsec-based host architecture for secure internet multicast', *Proceedings of the 7th Annual Network and Distributed System Security Symposium*, San Diego, California, USA, pp.23–39.

Casado, M., Freedman, M. J., Pettit, J., Luo, J., McKeown, N. and Shenker, S. (2007) 'Ethane: taking control of the enterprise', *SIGCOMM'07*, Kyoto, Japan, pp.1–12.

Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C. (2002) *Introduction to Algorithms*, 3rd ed., The MIT Press.

Hardjono, T. and Weis, B. (2004) *The Multicast Group Security Architecture*, RFC 3740 (Informational), IETF, March.

IEC TC 57/WG 10-12 (2003) 'IEC61850 communication networks and systems in substations', *IEC International Standard 61850*, April.

IEC TC 57/WG 10 (2010) 'Use Of IEC 61850 for the communication between substations', *IEC Technical Report 61850-90-1 Ed. 1.0*, March, pp.1–82.

IEC TC 57/WG 15 (2007) 'Power systems management and associated information exchange data and comm. security Part 6: security for IEC 61850', *IEC Technical Specification 62351-6 Ed. 1.0*, June, pp.1–15.

Igure, V.M., Laughtera, S.A. and Williamsa, R.D. (2006) 'Security issues in SCADA networks', *Computers & Security*, Vol. 25, No. 7, pp.498–506.

Liu, Y., Reiter, M.K. and Ning, P. (2009) 'False data injection attacks against state estimation in electric power grids', *the 16th ACM Conference on Computer and Communications Security (CCS)*, ACM, Chicago, IL, USA, pp.21–32.

Pannetrat, A. and Molva, R. (2003) 'Efficient multicast packet authentication', *Network and Distributed System Security Symposium*, San Diego, California, USA.

Park, J.M., Chong, E.K. and Siegel, H.J. (2002) 'Efficient multicast packet authentication using signature amortization', *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, California, USA, pp.227–240.

Perrig, A., Canetti, R., Tygar, J.D. and Song, D. (2000) 'Efficient authentication and signing of multicast streams over lossy channels', *Proceedings of IEEE Symposium on Security and Privacy*, Berkeley, California, USA, pp.56–73.

Ping Sun, J., Sheng, W., Wang, S. and Wu, K. (2002) 'Substation automation high speed network communication platform based on MMS+TCP/IP+Ethernet', *International Conference on Power System Technology*, Vol. 2, pp.1296–1300.

Skeie, T., Johannessen, S. and Brunner, C. (2002) 'Ethernet in substation automation', *IEEE Control Systems Magazine*, Vol. 22, No. 3, pp.43–51.

Smith, B.P. and Highfill, D.R. (2007) 'TVA investigates end-to-end integration', *Transmission and Distribution World*, VO. 59, No. 7, July, pp.20–26.

LAN/MAN Standards Committee of the IEEE Computer Society (2006) 'IEEE standard for local and metropolitan area networks-media access control (MAC) security', *IEEE Std 802.1AE-2006*, pp.1–42.

Substation Committee of the IEEE Power Engineering Society (2005) 'IEEE standard communication delivery time performance requirements for electric power substation automation', *IEEE Std 1646-2004*, pp.1–24.

Powner, D.A. and Rhodes, K.A. (2007) *Critical Infrastructure Protection: Multiple Efforts to Secure Control Systems Are Under Way, but Challenges Remain*, GAO Report to Congressional Requesters, US Government Accountability Office, September.

Wang, Q., Khurana, H., Huang, Y. and Nahrstedt, K. (2009) 'Time valid one-time signature for time-critical multicast data authentication', *IEEE Conference on Computer Communications (INFOCOM)*, Rio de Janeiro, Brazil, pp.1233–1241.

Wong, C.K. and Lam, S.S. (1999) 'Digital signatures for flows and multicasts', *IEEE/ACM Trans. Netw.*, Vol. 7, No. 4, pp.502–513.

Wu, F.F., Mosleshi, K. and Bose, A. (2005) 'Power system control centers: past, present, and future', *Proceedings of the IEEE*, Vol. 93, No. 11, November, pp.1890–1908.

Zhang, J. and Gunter, C.A. (2010) 'Application-aware secure multicast for power grid communications', *IEEE International Conference on Smart Grid Communications*, IEEE Communications Society, Gaithersburg, Maryland.

Zhang, J. (2010) *Secure Multicast for Power Grid Communications*, PhD Thesis, University of Illinois at Urbana-Champaign, Urbana, IL, USA.