SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2016; **9**:3602–3617
Published online 25 July 2016 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.1564

RESEARCH ARTICLE

A key management architecture and protocols for secure smart grid communications

Xuelian Long[1], David Tipper[1] and Yi Qian[2]*

[1] Graduate Telecommunications and Networking Program, University of Pittsburgh, Pittsburgh, PA, 15260, U.S.A.
[2] Department of Computer and Electronics Engineering, University of Nebraska—Lincoln, Lincoln, NE, U.S.A.

ABSTRACT

Providing encrypted communications among power grid components is expected to be a basic requirement of smart grid systems in the future. Here, we propose a key management architecture and associated protocols tailored to support encrypted smart grid communications. The architecture consists of two levels structured around the grid control system hierarchy. At the top level, which consist of control centers and regional coordinators, a bottom-up key structure is adopted using hash chaining and a logical key hierarchy. The lower level of the architecture consists of the regional coordinators (i.e., substations and distribution systems) and remote ends (e.g., meters and pole-top sensors) and utilizes a top-down key management approach built on an inverse element method. The proposed key management schema supports the hierarchical structure of the smart grid control mechanisms, and it takes the resource and electronic/physical security differences of the control levels into account. We define a set of protocols utilizing the architecture to provide secure unicast, multicast, and broadcast communications. Furthermore, we illustrate how the architecture is flexible enough to easily handle power grid nodes joining and leaving the system at the different levels. Lastly, we compare the proposed schema with existing ones and show that our architecture can achieve efficient key management to provide secure communications.

KEYWORDS
smart grid; key management; secure versatile communications

***Correspondence**
Yi Qian, Computer and Electronics Engineering, University of Nebraska—Lincoln, Lincoln, NE, U.S.A.
E-mail: yqian2@unl.edu

1. INTRODUCTION

The smart grid is envisioned as the next generation power grid that provides advanced electricity generation, distribution and management, utilizing the latest information and communication technologies to enable real-time load and control capabilities from the point of generation to the end user consumption point [1–3]. Providing reliable and secure communications among the grid components is considered a basic property that must be provided for the smart grid to function [2]. Appropriate security mechanisms such as encryption are essential to prevent inside and/or outside adversaries from forcing catastrophic decisions by modifying or forging collected data and control commands. A major challenge in supporting widespread encryption of smart grid communications is management of the encryption keys. In this paper, we focus on key management in order to provide secure smart grid communications†. Here, we attempt to design a scalable security architecture tailored to the power grid, particularly the grid control system and the unique aspects of the smart grid (e.g., variation in computational capabilities of devices and variety of communication patterns).

One of the biggest changes from the current power grid to the smart grid is the evolution of the control system from a centralized type of control to a hierarchical decentralized type of control [5–7]. Traditionally, the state of the power system is defined as the magnitude of the voltage and its phase angle at every bus in the system, and this is given as feedback to centralized controllers at the power companies and independent system operators (ISOs). In contrast, in the smart grid, high-precision measurement equipment and sensors are to be widely deployed allowing real-time determination of the demand, power quality, and fault diagnosis. In order to accommodate renewable distributed power generation sources and distributed energy storage options, the power control and balance will be implemented in a distributed hierarchical fashion. In such a hierarchical control system, the controllers at the top of the hierarchy take state input from lower layers and compute parameters that are passed to controllers at lower levels for their local control actions.

† An earlier version [4] of this paper was presented at the Fourth IEEE International Conference on Smart Grid Communications, Vancouver, Canada, 2013.

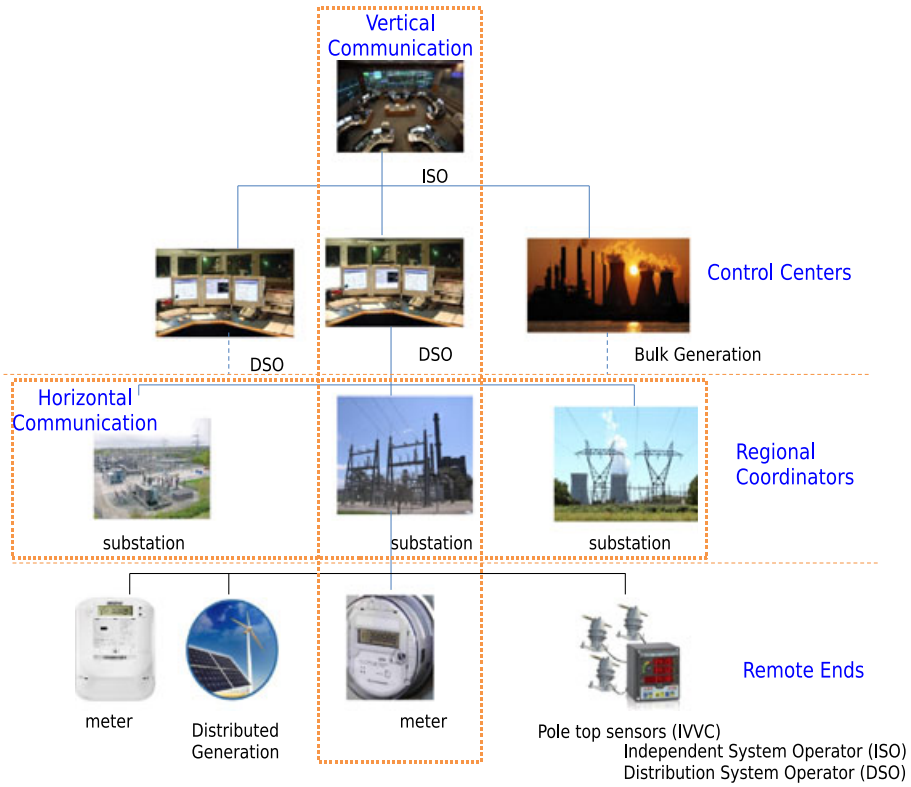

Figure 1. Grid control architecture.

Consider the high level hierarchical smart grid control architecture adapted from [6,7], shown in Figure 1. The components in the architecture can be categorized into three groups: (i) control centers (CCs), (ii) regional coordinators (RCs), and (iii) remote ends (REs). The control centers include ISOs, distributed system operators (DSOs), bulk power generation, and transmission systems. The RCs include the various substations and distribution systems. The REs consist of a variety of end devices, such as, smart meters, pole-top sensors, local distributed storage, local power generation, and supervisory control and data acquisition (SCADA) devices. Note, that the RE group includes a number of intelligent electrical devices such as voltage regulators, protective relays, and recloser controllers that contain low level microprocessors and have equipment lifetimes measured in decades. In contrast to the REs, the CCs and the regional centers usually have rich computational resources. Therefore, for communications between CCs and RCs, resource consumption is not the main issue. However, for communications with REs, resource consumption should be a major concern, especially the resource consumption of security mechanisms. In general, the communications between the control elements are time critical in nature implying the need for efficient algorithms that minimize the delay and computational costs.

In addition to the variations in computational resources, smart grid components will have large variations in their security environment. In North America, grid operators are expected to conform to the North American Electrical Reliability Corporation's Critical Infrastructure Protection Cyber Security standards [8,9], which require a number of protection measures in the bulk power generation and distribution system. For example, North American Electrical Reliability Corporation standards require electronic security perimeters (e.g., firewalls) and physical security controls around critical cyber assets (e.g., ISO control center). The standards classify cyber assets into three categories with low, medium, and high impact, which map into three levels of security controls. Hence, the CC level of the grid control architecture can be expected to have a strong security environment. Similarly, the RC level is expected to have an electronic security perimeter and physical security in place although at a somewhat weaker level than the CCs. However, the REs will typically have weak or no physical security and little or no electronic security perimeter. Thus, one can assume that REs maybe accessed by adversaries, and one must guard against stepping stone attacks launched from compromised REs.

Two types of communication are envisioned among the control architecture elements: vertical communication (e.g. from the ISO to REs) and horizontal peer-to-peer type of communication (e.g. among several substations under the same DSO). Both types of communication may use broadcast communication mode (e.g., from a DSO to all REs belonging to the DSO), multicast communication mode (e.g., among several RCs), or unicast communication mode

(e.g., from a ISO to a DSO). Therefore, the security architecture must be versatile enough to support the various communication modes.

Note that power grid nodes (CC, RC, and RE) in the control architecture may leave the architecture because of failures, scheduled downtime (e.g., for upgrades/patches), loss of customers, etc. Similarly, nodes may join the grid for a variety of reasons, such as expansion and addition of customers. Hence, the security architecture must be flexible enough to support management of the keys for nodes joining the grid and nodes leaving the grid. One would expect that nodes joining/leaving will happen infrequently in the level of CCs and RCs with the primary driver being scheduled downtime, failures, accidents, and acts of nature (e.g., hurricane and ice storm), whereas at the RE level, the components can be expected to come and go more frequently because of power customer churn and both accidental failures and natural hazards. Because the number of grid components will change over time with devices being added and removed, the security architecture should support backward and forward secrecy. Backward secrecy indicates that the future secret keys must inaccessible by former entities. Forward secrecy implies that the previous used secret keys must be inaccessible by new grid entities.

Here, we propose a novel smart grid key management architecture that is structured in a multi-level fashion similar to the power grid control architecture. Specifically, there are two levels to the security architecture with the top level consisting of the control centers and regional coordinators, whereas the bottom level is composed of the regional coordinators and remote end units. Two different key management mechanisms are introduced for upper level communications and lower level communications, respectively. The two different key management mechanisms are adopted because of the variations in the electronic/physical security environment and computational resources of the devices in the two levels. At the top level, our schema employs the Iolus framework [10] for structured key management. In the Iolus framework, the membership changes in a subgroup will not affect other subgroup members; thus, it mitigates the impacts of changes on the whole system. At the bottom level, the key management is constructed using a inverse element approach. The proposed key management schema is a versatile schema as defined in [11], in that it supports secure broadcast, multicast, and unicast communications. Further, the architecture provides backward and forward secrecy. We develop protocols for our architecture to support both horizontal and vertical communications in multicast, broadcast, and unicast communication modes. A comparative analysis of the proposed architecture with existing schemas [11,12], shows that our approach requires less communication and storage costs and it can support the hierarchical structure of the smart grid control system while incorporating the variations in resources among components.

The rest of this paper is organized as follows. Section 2 discusses related work. The key management architecture is presented in Section 3. The protocols to make use of the architecture in communications is shown in Section 4. Section 5 describes how the dynamics of nodes joining and leaving the key management architecture are handled. Section 6 gives the security analysis and comparison with some existing key management schemas. Conclusions are given in Section 7.

2. RELATED WORK

Here, we highlight the research literature that is relevant to the development of our proposed security architecture, and a broader recent survey of cyber security literature for Smart Grid systems is given in [2]. The studies on determining trust models can be helpful for securing communications among various devices. These trust-based approaches can be classified into several categories, such as authentication management for public keys [13], e-commerce [14], and peer-to-peer networks [15]. This includes work on key management schemas for secure communications. Because the key management of multicast communications is more complex than unicast communications [16], a key tree management structure has been widely discussed for multicast communications. Usually, these methods adopt a hierarchy of keys for multicast communications, in which each member is assigned a set of keys based on its location in the key tree [17,18]. In wireless sensor networks, Ren *et al.* proposed an efficient key generation schema, which could greatly reduce the storage cost of pre-key distribution [19]. In [20], Ren *et al.* proposed three different broadcast authentication mechanisms in wireless sensor networks and quantitatively analyze their energy consumption. However, they did not discuss unicast authentication. Law *et al*. [21] proposed WAKE, which is a key management schema for wide-area measurement system in smart grid. This schema is based on multiple-time signature schemas, which is different from our binary tree-based approach. The work of Nicanfar *et al.* [22] is focused on the smart grid mutual authentication and a smart grid key management protocol.

The Internet Engineering Task Force (IETF) Multicast Security Framework [23] consists of several elements including a group controller and key server (GCKS), group key management, and group security associations in which the GCKS is a centralized host. To provide scalable multicast key management for heterogeneous wireless networks with high link error rates, Sun *et al.* [24] proposed a method to manage multicast key trees that match the network topology and thus reduced the communication overhead of rekeying. In [25], Kong *et al*. proposed a *k-hop* clustering-based approach to construct a backbone of GCKS nodes, in which each GCKS is also the leader of its cluster. It indicates that *k-hop* clustering reduces the number of GCKS nodes and security vulnerabilities. However, Kong *et al*.[25] do not depict the algorithm about electing the cluster leader. Choi *et al.* [12] proposed a key management schema for secure group communications in SCADA systems. It uses the Iolus framework to reduce the effects of group membership

DOI: 10.1002/sec

changes to the other groups, and it could also support unicast communications and broadcast communications in SCADA. However, it uses the same key management mechanisms at the remote meters as that in the sub-master terminal units, which ignores the limited resources issue at the remote meters. Furthermore, by using the same key structure throughout the system, it ignores the variations in electronic and physical security of the devices. Kim and Choi [11] proposed a more efficient schema than [12] for secure communications in smart grid systems, which could greatly reduce the resource consumption for key updates. However, it does not adopt a hierarchical structure. The recent work by Cao *et.al.* [26] discussed the need for a layerd security architecture incorporating the variations in computational capabilities and physical and electronic security of devices in critical infrastructures such as the power grid.

3. KEY MANAGEMENT ARCHITECTURE

In this section, we first present motivation of the proposed schema and then present our hierarchical key management architecture.

3.1. Motivation

For providing secure communication for distributed smart grid systems, security and efficiency are two of the most

Table I. Summary of notation.

Notation	Definition/description
CC_i	The ith control center
RC_i	The ith regional center
RE_i	The ith remote endpoint
$h_{CC,RC}$	Height of a tree between a CC and RCs
$h_{RC,RE}$	Height of a tree between a RC and REs
q	Number of RCs
r_i	Number of REs under RC_i
KDC	Key distribution center
K_{CC_i}	The key of CC_i
$K_{i,j}$	The jth key at level i in a binary tree
$K^x_{i,j}$	The jth key at level i in a binary tree in region x
k_i^{boot}	The bootstrap key of device i
$E_K()$	Encryption using key K
E_{data}	Encrypted message/cipher text
$D_K()$	Decryption using key K
$Sign_t$	Signature created at transmitter side
$Sign_r$	Signature created at receiver side
$SigID_i$	Signature ID of node i
$H()$, $HL()$, $HR()$	Different hash functions
SID	Session identity
K_{REGION}	The broadcast key in a region
$Node_{i,j}$	The jth node at level i
$Node_j$	The jth node in a non-hierarchical structure
$S_{i,j}$	Secret value of the jth node at level i
S_j	Secret value of the jth node in a non-hierarchical structure
SET_{RE_i}	Set of secret values that RE_i possesses
$K_{session}$	Session key
$H_K()$	A keyed-hash function using K as the key
K_{gs}	Group session key
N_g	Set of multicast communication group nodes
K_{bs}	The broadcast session key
GM_K	Set of group members that share the key K
AK_K	Set of ancestor keys of K, except for parent of K

CC, control center; RE, remote end; RC, regional coordinator.

import aspects. That is, the proposed security architecture should support backward and forward secrecy, as well as consume as little resources as possible. In the current smart grid systems, there are limited resources at remote meters but such a condition is ignored in the current researches (e.g., [12]). Thus, our proposed key management architecture aims to provide a more efficient key management schema for the distributed system and at the same time providing backward and forward secrecy.

3.2. Proposed key management schema

We list the definition/description of the symbols/notations used in the paper in Table I. We show our key management architecture in Figure 2. The architecture contains three classes of components: CCs, RCs, and REs. The upper level of the key structure (i.e. between the CC and RCs) is constructed based on the Iolus framework [10] and logical key hierarchy (LKH) using a bottom-up approach [17]. In the upper level, the CC acts as a key distribution center (KDC). The lower level of the key structure (i.e. between RCs and REs) is constructed using a top-down approach based on an inverse element structure. In the lower level, RCs act as KDC for the attached REs. Note that an RC contains two types of keys, one for the upper level and the other for the lower level. Thus, the RC may contain two sets of encryption and hash algorithms one for the upper level and the other for the lower level.

3.2.1. Key structure in the upper level.

In the upper level, we utilize the Iolus framework, which was proposed for secure multicasting based on a secure distribution tree. The key structure in the upper level employs a binary tree and LKH. A CC is the root node of a binary tree, and RCs are leaf nodes of the binary tree. The binary tree as shown in Figure 2 contains q leaf nodes. Iolus uses a group security controller (GSC) to manage the keys in a KDC fashion. Here, the CC acts as the GSC and KDC. The height of the tree is $h_{CC,RC} = log_2(q)$, if the tree is complete, whereas the height of an incomplete tree is $h_{CC,RC} = \lceil log_2(q) \rceil$. Each RC leaf node will be assigned a secret key when it joins the system, and for leaf node i, the secret key is $K_{h_{CC,RC},i}$ The secret keys are assigned using the bootstrap procedure discussed in Section V. A form of hash chaining is used to determing the keys as one moves up the tree. Specifically, a parent node secret key is constructed in a bottom-up fashion by hashing the secret keys of its two children nodes, as defined in Equation (1):

$$K_{h_{CC,RC}-1-j,(i+1)/2} = H\left(H(K_{h_{CC,RC}-j}, i), H(K_{h_{CC,RC}-j,i+1})\right) \quad (1)$$

where H is a hash function, i is an odd number, $1 \leq i < 2^{h_{CC,RC}-j}$, and $0 \leq j < h_{CC,RC}$. If a node has only one child, then it creates a virtual copy of itself as a child and obtains a leaf node key for the virtual copy, which is then used in Equation (1) with the single child key. Scalability is provided by designating some nodes in the tree as group security intermediaries (GSI), thereby breaking up the tree into subgroups. The GSC manages the top-level subgroup and GSIs. The GSC knows the keys of the GSIs, and a GSI knows all the keys and group key of the nodes that belong to the GSI.

3.2.2. Key structure in the lower level.

The lower level uses a different key structure from the upper level. This is intentional to accommodate the computational resource constraints and weaker physical/electronic security of REs. One difference from the key structure in the upper level is that there is a region key K_{REGION_m} (i.e., group key) for all the REs under RC_m. The lower level employs a top-down inverse element approach for the sub-binary tree that contains one RC and all REs logically belonging to the RC. The RC acts as a KDC and will assign the corresponding secret keys obtained through the inverse element approach and the region key to all other nodes in the sub-binary tree. Assume RC_m has r

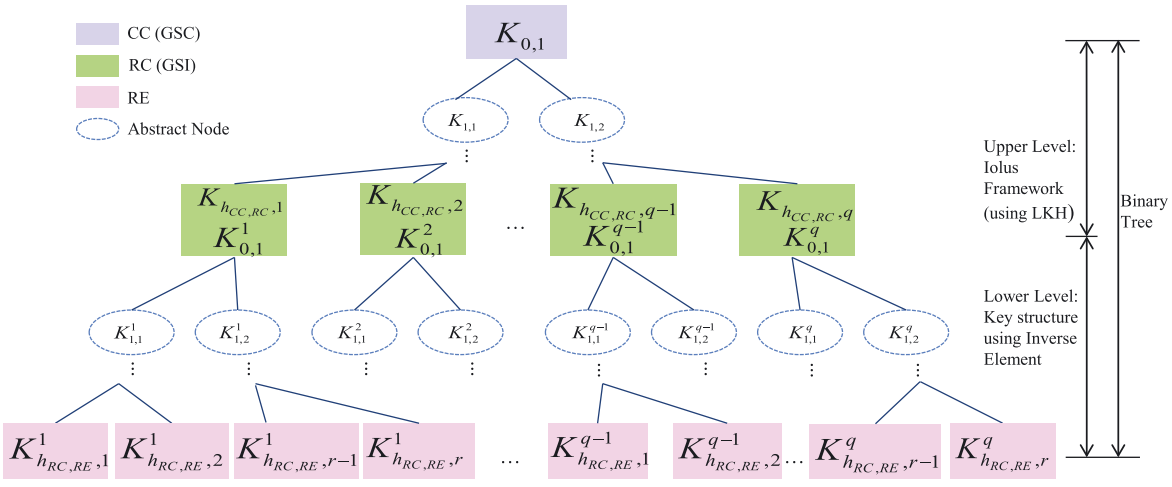

Figure 2. The proposed key management architecture. CC, control center; RE, remote end; RC, regional coordinator.

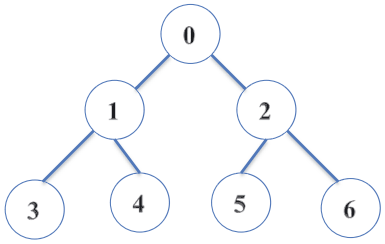

Figure 3. Key structure in lower level using inverse elements.

REs. First RC_m will be assigned a secret value $S_{0,1}$ as its secret key $K_{0,1}^m$. The secret values of its children nodes are constructed based on its own secret value, using:

$$\begin{cases} \mathbf{S}_{i,j} = \boldsymbol{HL}(\boldsymbol{S}_{i-1,(j+1)/2}) \\ \mathbf{S}_{i,j+1} = \boldsymbol{HR}(\boldsymbol{S}_{i-1,(j+1)/2}) \end{cases} \tag{2}$$

where j is an odd number, $HL()$ is the hash function for the left child, and $HR()$ denotes the hash function for the right child. For a non-root node of the binary tree in the lower level, the secret value is not its secret key. Before presenting the secret key of non-root nodes, we first introduce the concept of a restricted path for a non-root node. The restricted path of a non-root node is the path from itself to the root node in the binary tree. The secret key of a non-root node is the set of all the secret values in the tree except secret values along the restricted path. For example, in Figure 3, the restricted path of $Node_3$ is $Node_3$—$Node_1$—$Node_0$. Thus, the secret key of $Node_3$ is all other secret values in the tree except the secret values of $Node_3, Node_1$, and $Node_0$, which are the secret values of $Node_4$,, $Node_2$, $Node_5$ and $Node_6$. Because the secret values of $Node_5$ and $Node_6$ can be calculated based on the secret value of $Node_2$ using Equation (2), the secret key of $Node_3$ is as follows:

$$\begin{aligned} K_3 = \boldsymbol{SET}_3 &= \{\overline{\boldsymbol{S}_3, \boldsymbol{S}_1, \boldsymbol{S}_0}\} \\ &= \{S_4, S_2, S_5, S_6\} \\ &= \{S_4, S_2\} \end{aligned} \tag{3}$$

The last step is obtained because $\boldsymbol{S}_5 = \boldsymbol{HL}(\boldsymbol{S}_2)$ and $\boldsymbol{S}_6 = \boldsymbol{HR}(\boldsymbol{S}_2)$. In this way, we can obtain secret keys of all the nodes in the binary tree and the root-node RC knows all the secret keys. Note that, every non-root node (i.e. RE) knows its own key and the region key only.

4. KEYS AND PROTOCOLS FOR VERSATILE COMMUNICATIONS

In this section, we present protocols illustrating how to use the proposed key management schema in unicast, multicast, and broadcast communications. We do not specify the encryption algorithms or hash functions to be used but note that the computational differences between the levels in the security architecture should be considered in their selection. We assume that the communication network interconnecting the power grid components meets recommended US Department of Energy Quality of Service goals [27], such that delay, packet loss, and availability are not an issue. Further, we assume all devices (REs, RCs, and CCs) are addressable with MAC and/or IP addresses.

We assume that an adversary has one or more of the following goals: (i) to inject a counterfeit a message, (ii) to modify an existing message to one or more receivers, and (iii) to passively collect information from messages. Furthermore, we assume that the adversary has the following capabilities: access to all communication links, the ability to capture, resend or eavesdrop on some or all packets, and much greater computation power than the normal network components (i.e., RCs and REs).

In order to provide secure communications, we adopt the use of one time session keys $K_{session}$, which are used to guarantee the freshness of the key and the confidentiality of the communication by encryption. A session identity (SID) is used as an input to the session key generation. The SID is a random number, which is used to guard against a replay attack. Additionally, we adopt the use of signatures $Sign_t$ and $Sign_r$, which are the signatures of the messages computed at the transmitter side and receiver side, respectively. If the signatures are the same when compared at the receiver, the integrity of the communication is guaranteed. Also each node i has a signature ID $SigID_i$ derived via a hash function of the node's network address.

4.1. Unicast communications

Unicast communications will be used in a number of scenarios, and Table II summarizes the session keys for unicast communications. We first demonstrate the process of secure unicast communications between grid components in the same layer of the security architecture, for example, between a CC and a RC, or a RC and a RE underneath it. Figure 4 illustrates the communication process that has the three steps given as follows.

Step 1 N_i generates session key $K_{session}$
$E_{data} = E_{K_{session}}(\boldsymbol{Data})$
$\boldsymbol{Sign}_t = H_{K_{session}}(E_{data})$
Step 2 $N_i \longrightarrow N_j$: $(E_{data} \parallel \boldsymbol{Sign}_t \parallel \boldsymbol{SID})$.
Step 3 N_j generates session key $K_{session}$
$\boldsymbol{Sign}_r = H_{K_{session}}(E_{data})$
IF $\boldsymbol{Sign}_r = \boldsymbol{Sign}_t$
$\boldsymbol{data} = D_{K_{session}}(E_{data})$
END

Step 1 in the procedure involves determining the session key, encrypting the data, and then determining the integrity signature by hashing the encrypted data. For unicast communication between a RC and a RE, the session key can be obtained by hashing the secret set of the RE and the SID. For unicast communications between two nodes in the upper level, the session key is to hash the shared key (K_{share}) between two nodes through the logical key

Table II. Keys for unicast communications.

Unicast scenario	Session key
A RC and a RE	$K_{session} = H(SET_{RE} \parallel SID)$
A RE and a RE under the same RC	$K_{session} = H(\oplus(\bigcap SET_{\{REs\}}) \parallel SID)$
A RC and a RC	$K_{session} = H(K_{share} \parallel SigID_i \parallel SigID_j \parallel SID)$
A CC and a RC	$K_{session} = H(K_{RC} \parallel SigID_{CC} \parallel SigID_{RC} \parallel SID)$

CC, control center; RE, remote end; RC, regional coordinator; SID, session identity.

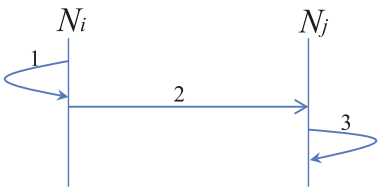

Figure 4. Process of unicast communications in upper level or in the same region in the lower level.

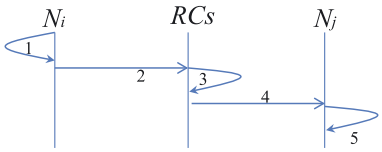

Figure 5. Process of unicast communications that depends on the relay of regional coordinator (RC)(s).

hierarchy structure, together with the signature of the two nodes and a *SID*. For example, the shared key between the CC and a RC is K_{RC}, and the session key generation is shown in the fourth row of Table II.

If two REs under the same RC would like to communicate with each other, the session key is obtained by hashing the result of XORing the elements in the intersection (i.e., $\oplus(\bigcap SET_{\{REs\}})$) of the two REs' secret sets and a SID. For example, the session key between $Node_3$ and $Node_6$ in Figure 3 is $K_{session} = H(S_4 \oplus S_{Node_5} \parallel SID)$. This is because $K_3 = \{S_2, S_4, S_5, S_6\}$ and $K_6 = \{S_1, S_3, S_4, S_5\}$, so the intersection of K_3 and K_6 is $\{S_4, S_5\}$.

Note, that there are unicast communication cases that depend on the relay of a RC or RCs, as shown in Figure 5. Examples of such communication include unicast between a CC and a RE and unicast between REs that are attached to different RCs. The steps involved for communication are as follows.

Step 1 N_i generates session key $K_{session}$
$E_{data} = E_{K_{session}}(Data)$
$Sign_t = H_{K_{session}}(E_{data})$
Encrypt session key: $E_{K_{N_i,RC}}(K_{session})$

Step 2 $N_i \longrightarrow RC$: $(E_{data} \parallel Sign_t \parallel E_{K_{N_i,RC}}(K_{session}) \parallel SID)$.

Step 3 RC decrypts session key $K_{session} = D_{K_{N_i,RC}}(E_{K_{N_i,RC}}(K_{session}))$ and encrypts it with the key known to both RC and node N_j, that is, $E_{K_{RC,N_j}}(K_{session})$. If there are two RCs involved in the communications, there will be another round of the decryption and encryption procedure.

Step 4 $RC \longrightarrow N_j$: $(E_{data} \parallel Sign_t \parallel E_{K_{RC,N_j}}(K_{session}) \parallel SID)$.

Step 5 N_j decrypts session key and data
$K_{session} = D_{K_{RC,N_j}}(E_{K_{RC,N_j}}(K_{session}))$
$Sign_r = H_{K_{session}}(E_{data})$
IF $Sign_r = Sign_t$
$data = D_{K_{session}}(E_{data})$
END

Here, in Step 1, the sender node N_i also encrypts the session key using the key $K_{N_i,RC}$, which is known by the RC and itself. If the sender node is a CC, then $K_{N_i,RC} = K_{RC}$; if the sender node is a RE, then $K_{N_i,RC} = SET_{RE}$. At Step 2, the sender node sends the encrypted message, signature, the encrypted session key, and *SID* to the RC. In Step 3, one or more RCs will facilitate the communication between N_i and N_j. Consider the case where there is only a single RC acting as an intermediate node; the RC decrypts the session key using the key known to the RC and the sender node N_i and then encrypts the session key by the key known only to the receiver node N_i and itself. In Step 4, the RC sends the encrypted message, signature, the encrypted session key, and *SID* to the receiver. In Step 5, the receiver first decrypts the session key, then computes the signature, and compares it with the received signature. If they are the same, the receiver will decrypt the cipher text to obtain the message.

4.2. Multicast communications

Multicast communications could happen between a node N_i and a group of nodes N_g. For example, between a CC and several RCs, a RC and several REs, as well as a CC and multiple REs. A multicast group session key K_{gs} is utilized in multicast communications and can be obtained by hashing a shared key (K_{share}) among the group of nodes and a *SID*. As noted earlier, the upper level key structure is based on the Iolus framework, which was developed specifically for scalable secure multicast communications [10]. In the framework, the GSC and GSI maintains access control lists of multicast groups each with a common shared key K_{share} as discussed in [10]. In the lower level, when a RC multicasts a message to many REs in the same region, the group

session key is obtained by hashing the result of XORing the elements in the intersection (i.e., $\oplus(\bigcap SET_{\{RES\}})$) of these REs' secret sets and a *SID*. Table III summarizes the session keys for multicast communications. The process in upper level/lower level is shown in Figure 6 with the corresponding steps given as follows.

Step 1 N_i computes group session key K_{gs}
$E_{data} = E_{K_{gs}}(Data)$
$Sign_t = H_{K_{gs}}(E_{data})$
Step 2 $N_i \longrightarrow N_j$: $(E_{data} \parallel Sign_t \parallel SID), N_j \in$multicast group N_g, which know K_{gs}.
Step 3 N_j computes session key K_{gs}
$Sign_r = H_{K_{gs}}(E_{data})$
IF $Sign_r = Sign_t$
$data = D_{K_{gs}}(E_{data})$
END

In Step 1, the sender computes the group session key K_{gs}, then encrypts the data, and computes a message signature. As shown in Step 2, the source multicasts the cipher text, signature, and *SID* to all the group member nodes. Each group member receives the message, computes the group session key, verifies the signature, and then obtains the plain text as shown in Step 3.

In a fashion analogous to unicast communications, there are cases where multiple RCs are needed to relay the communications. In particular, the case of multicast communications from the CC to multiple REs depends on the relay of the RC(s) that control the REs. Also, if a RE would like to multicast a message to a group of REs under a different RC, two RCs act to help exchange the session key.

Table III. Keys for multicast communications.

Multicast scenario	Session key
A CC/RC to many RCs	$K_{gs} = H(K_{share} \parallel SID)$
A RC/RE to many REs	$K_{gs} = H(\oplus(\bigcap SET_{\{REs\}}) \parallel SID)$

CC, control center; RE, remote end; RC, regional coordinator; SID, session identity.

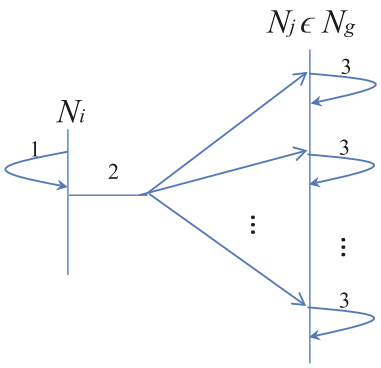

Figure 6. Process of multicast communications in upper or lower level.

That is, the first RC, which the transmitter RE belongs to, encrypts the session key with a key that the second RC (the receiver group belongs too) knows; thus, the second RC can decrypt the session key and encrypt it with the shared key of the receiver group. Then, the receiver group of REs can get the session key.

4.3. Broadcast communications

Broadcast communications can occur between a CC and all RCs in the upper layer, between a RC and all REs under the RC in the lower layer and accross the two layers between a CC to all REs. Table IV lists the session key generation procedure for broadcast communications. The most complicated case is broadcast communications between a CC and all REs, which requires all the RCs to serve as relays. Figure 7 shows the process for the CC to REs broadcast case. The steps corresponding to Figure 7 are listed as follows.

Step 1 CC computes broadcast session key K_{bs}
$E_{data} = E_{K_{bs}}(Data)$
$Sign_t = H_{K_{bs}}(E_{data})$
Encrypt session key: $E_{K_{CC}}(K_{bs})$
Step 2 CC $\longrightarrow$ RCs: $(E_{data} \parallel Sign_t \parallel E_{K_{CC}}(K_{bs}) \parallel SID)$
Step 3 RC decrypts the broadcast session key as it knows K_{CC} and encrypts the broadcast session key with the region key: $E_{K_{REGION}}(K_{bs})$
Step 4 RC $\longrightarrow$ REs: $(E_{data} \parallel Sign_t \parallel E_{K_{REGION}}(K_{bs}) \parallel SID)$
Step 5 RE decrypts broadcast session key
$K_{bs} = D_{K_{REGION}}(E_{K_{REGION}}(K_{bs}))$
$Sign_r = H_{K_{bs}}(E_{data})$
IF $Sign_r = Sign_t$
$data = D_{K_{bs}}(E_{data})$
END

Table IV. Keys for broadcast communications.

Broadcast Scenario	Session Key
A CC to all RCs or to all REs	$K_{bs} = H(K_{CC} \parallel SID)$
A RC to all REs	$K_{bs} = H(K_{REGION} \parallel SID)$

CC, control center; RE, remote end; RC, regional coordinator; SID, session identity.

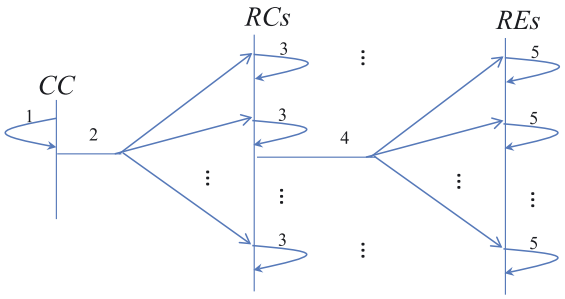

Figure 7. Process of broadcast communications between a control center (CC) and all remote ends (REs). RCs, regional coordinators.

The CC first computes a broadcast session key K_{bs} and uses this key to encrypt data and generate the signature.

If the broadcast communications only happen between a CC and all RCs or a RC and all REs under the RC, the broadcast session keys are generated according to Table IV. If the broadcast communications happen between a CC and all REs, as shown Figure 7, the CC first generates the broadcast session key K_{bs}, encrypts the data, computes the signature, and encrypts the broadcast session key with its own key K_{CC}. Then the CC broadcasts the cipher text, signature, encrypted broadcast session key, and *SID* to all RCs. A RC decrypts the session key, re-encrypts it with the corresponding region key, and then broadcasts the previous cipher text, signature, the new encrypted session, and *SID* to all REs under itself. At last, REs decrypt the broadcast session key, verify the signature, and then obtain the plain text.

5. HANDLING GRID DYNAMICS

As noted earlier, power grid nodes (CC, RC, and RE) in the key management architecture may leave the architecture because of failures, scheduled downtime (e.g., for upgrades/patches), loss of customers, etc. Similarly, nodes may join the grid for a variety of reasons, such as expansion and addition of customers. Hence, the architecture must be flexible enough to support management of the keys for nodes joining and leaving the grid. One would expect that nodes joining/leaving will happen infrequently in the upper level of CCs and RCs, whereas in the lower level, REs can be expected to come and go more frequently.

5.1. Nodes joining

In order for nodes to join the system, a bootstrapping procedure is needed to identify the joining device to the appropriate KDC and invoke the joining protocol. We assume that each device RC_i or RE_i comes with a factory printed bootstrap key K_i^{boot} or one is installed by the operator (e.g., 192-bit Advanced Encryption Standard (AES) key). At the time of installation, a technician enters the device's bootstrap key into the appropriate KDC; that is, if the device is an RC, then the bootstrap key is added to the KDC at the CC, whereas if the device is an RE, then the bootstrap key is added to the KDC at its local RC. Once connected to the network, the RC_i or RE_i sends a K_i^{boot} encrypted join request to appropriate KDC to initiate the join protocol as discussed for each level in turn later.

5.1.1. Join protocol for upper level.

If a new RC_i node would like to join the system, it should follow the bootstrap procedure to send a join request to the KDC at the CC as follows.

Step 1 RC_i sends join request to KDC
$K_i^{bootsession} = H(K_i^{boot} \parallel SID)$
$E_{data} = E_{K_i^{bootsession}}(Data)$
$Sign_t = H_{K_i^{bootsession}}(E_{data})$
$RC_i \longrightarrow$ KDC at CC: $(SigID_{RC_i} \parallel E_{data} \parallel Sign_t \parallel SID)$

Step 2 KDC at CC decrypts join request
$K_i^{bootsession} = H(K_i^{boot} \parallel SID)$
$Sign_r = H_{K_i^{bootsession}}(E_{data})$
IF $Sign_r = Sign_t$
$Data = D_{K_i^{bootsession}}(E_{data})$
END

Upon receiving a join request, the CC must add the new RC_i node to the binary tree and adjust the keys in the tree as necessary. At the upper level, the key structure works from the leaf nodes up using a hashing of left and right children

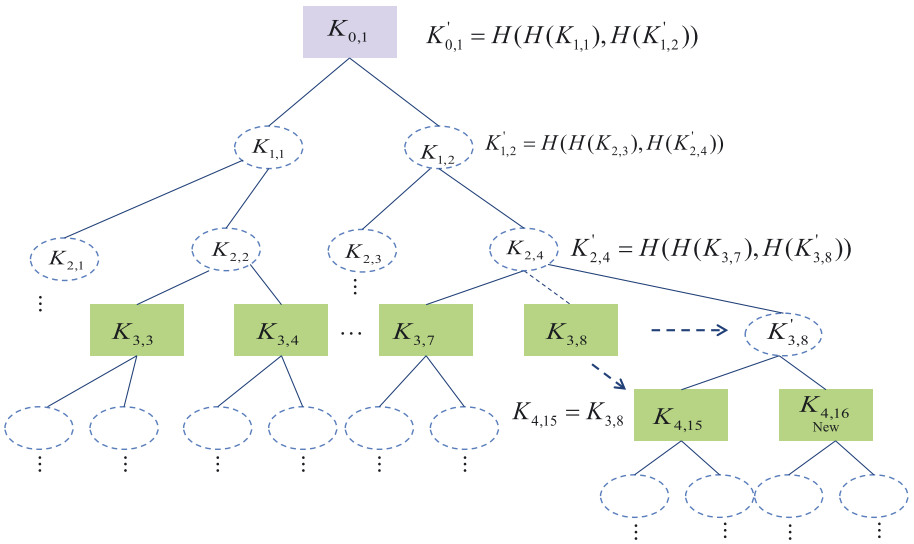

Figure 8. Node joining phase in the upper level.

DOI: 10.1002/sec

keys, and a full binary tree is required (i.e., every non-leaf node must have two children). In order to add a new node in to the full binary tree, one needs to expand the tree by taking a leaf node j creating a virtual copy of the node to replace itself as a parent node, inserting itself (e.g., the original leaf node) as the left child while adding the new RC_i node as the right child node. The joining node is given a new key, while the secret key of the node that has been moved to the child position is unchanged (i.e., the original leaf node), but the keys of the parent and ancestor nodes should be updated using Equation (1). In Figure 8, we give an example of the process. Because the binary tree is full, a virtual node $Node'_{3,8}$ is created to replace the previous leaf node $Node_{3,8}$. And the previous leaf node $Node_{3,8}$ is move down as the left child node $Node_{4,15}$ of the virtual node $Node'_{3,8}$. The new joining node is assigned position $Node_{4,16}$ as the right child node of $Node'_{3,8}$. $Node_{4,16}$ is assigned $K_{4,16}$ by the KDC, while the left child $Node_{4,15}$ retains the previous key, which is relabeled (i.e., $K_{4,15} = K_{3,8}$). Next, the keys of $Node_{3,8}$, $Node_{2,4}$, $Node_{1,2}$, and $Node_{0,1}$ should be updated using Equation (1), as shown in Figure 8.

After the key updates, the change of keys should be propagated to other nodes. That is, a node whose key changed encrypts its new key and the new keys of its ancestor nodes using its left child node's key and then propagates the encrypted message to its left sub-binary tree. For example, $Node_{2,4}$, $Node_{1,2}$, and $Node_{0,1}$ need to propagate the changes to their left children. This can be accomplished by a targeted multicast from the nodes in question. Finally, the KDC will encrypt all the keys of the nodes on the restricted path of the new joining node and send this information to the joining node. Hence, the join protocol after Steps 1 and 2 earlier is as follows:

Step 3 KDC at CC indicates to $Node_{h,m}$ to create a virtual node $Node'_{h,m}$ for itself and move itself to the position of the left child $Node_{h+1,2m-1}$ without changing its key $K_{h+1,2m-1} = K_{h,m}$.

Step 4 KDC at CC generates and distributes a new key $K_{h+1,2m}$ for the joining RC_i node, which will be inserted to the key structure as $Node_{h+1,2m}$
$K_i^{bootsession} = H(K_i^{boot} \parallel SID)$
$E_{data} = E_{K_i^{bootsession}}(K_{h+1,\,2m})$
$Sign_t = H_{K_i^{boot}}(E_{data})$
KDC at CC $\longrightarrow RC_i$ ($SigID_{CC_i} \parallel E_{data} \parallel Sign_t \parallel SID$)

Step 5 RC_i obtains its secret key
$K_i^{bootsession} = H(K_i^{boot} \parallel SID)$
$Sign_r = H_{K_i^{bootsession}}(E_{data})$
IF $Sign_r = Sign_t$
$K_{h+1,2m} = D_{K_i^{bootsession}}(E_{data})$
END

Step 6 The parent and ancestor keys $K_{i,j} (0 \leq i \leq h, j = \lceil \frac{m}{2^{h-i}} \rceil)$ are updated as $K'_{i,j} = H(H(K_{i+1,2j-1}), H(K_{i+1,2j})))$ at the nodes along the restricted path and in the KDC.

Step 7 The ancestor nodes with updated keys propagate the updated keys as needed using multicast communications.

Step 8 KDC at CC sends the keys $K'_{i,j} (0 \leq i \leq h, j = \lceil \frac{m}{2^{h-i}} \rceil)$ to the new joining RC_i using unicast communications.

5.1.2. Join protocol for lower level.

At the lower level, each RC_i acts as a KDC for the REs attached to it. Consider a new RE_j sending a join message to RC_q to which it is attached following a bootstrap procedure similar to the upper level. The steps are as follows.

Step 1 RE_j sends join request to KDC at RC_q
$K_j^{bootsession} = H(K_j^{boot} \parallel SID)$
$E_{data} = E_{K_j^{bootsession}}(Data)$
$Sign_t = H_{K_j^{bootsession}}(E_{data})$
$RE_j \longrightarrow$ KDC at RC_q: ($SigID_{RE_j} \parallel E_{data} \parallel Sign_t \parallel SID$)

Step 2 KDC at RC_q decrypts join request
$K_j^{bootsession} = H(K_j^{boot} \parallel SID)$
$Sign_r = H_{K_j^{bootsession}}(E_{data})$
IF $Sign_r = Sign_t$
$Data = D_{K_j^{bootsession}}(E_{data})$
END

After processing the join request, RC_q decides the position (l, k) where the new node will be inserted into the binary tree structure (usually a leaf node position). If the binary tree is perfect before the new RE_j joins the tree, the RC will expand the the binary tree to another layer by moving the leave nodes as the left children nodes at the new level, then inserting the new RE_j as a right child node at a new level in the tree. The key structure will be updated for the remaining REs based on the following protocol:

Step 3 RC_q broadcasts a join message $E_{K_{REGION}}(JOIN, ENTITY_{index}, (l, k))$ in the region.

Step 4 If the new node has a sibling in the expanded tree, the sibling will obtain $S_{l,k}$ from the RC.

Step 5 All nodes update the new structure of the binary tree and locally update the region key by $K'_{REGION} = H(K_{REGION} \oplus S_{l,k})$ where $\oplus$ stands for the XOR function.

Step 6 RC_q uses a bootstrap-based session key to send the joining RE_j node K'_{REGION}, $SET_{l,k}$ and its position (l, k) in the tree.

We use an example to show this process. As shown in Figure 9, a new node would like to join the system, and after the bootstrap based joins request (Steps 1 and 2), the RC_q decides to assign it to $Node^q_{3,8}$ in the key structure. Next the following steps are executed.

Step 3 RC broadcasts $E_{K_{REGION}}(JOIN, ENTITY_8, (3, 8))$.

Step 4 $Node^q_{3,7}$ will obtain $S_{3,8}$ from the RC.

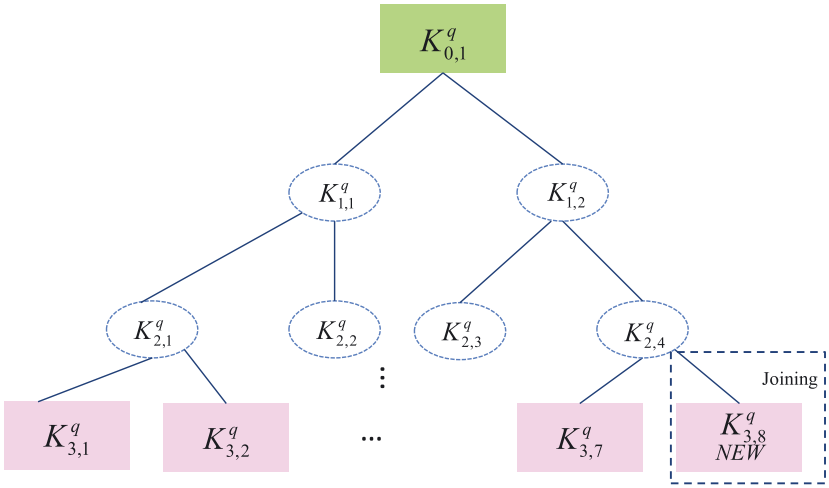

Figure 9. Node joining phase in the lower level.

Step 5 All nodes update the region key by $K'_{REGION} = H(K_{REGION} \oplus S_{3,8})$ and $S_{3,8} = HR(S_{2,4})$.

Step 6 RC uses a bootstrap-based session key to send $K_{3,8} = \{S_{3,7}, S_{2,3}, S_{1,1}\}$, K'_{REGION} and its position in the key structure to the joining RE_j ($Node^q_{3,8}$).

5.2. Nodes leaving

Similar to nodes joining the system, the keys must be managed when a node departs the system. A node departure can be invoked either by a KDC at the appropriate level in the hierarchy or by the node itself. Once the leave message is authenticated and acknowledged, the KDC must remove the departing node from the bootstrap database and the key tree structure, and then the tree must be rearranged and updated. We consider a more detailed description for each level in turn.

5.2.1. Leave protocol for upper level.

When a RC_i node leaves the system, the KDC will remove it from the binary tree, and its sibling node will be moved up to the position of the parent node. Once the tree is rearranged, the keys along the restricted path are updated using Equation (1), and the information is propagated as needed. The leaving protocol is disscussed as follows.

Step 1 The KDC at the CC or the RC_i uses the unicast communication protocol to issue the *Leave* command.

Step 2 The recipient of the *Leave* command (KDC at CC or RC_i) sends an acknowledgement using the unicast communication protocol, and the KDC at the CC receives the KDC key information for the lower level nodes attached to RC_i.

Step 3 The KDC examines the position of $Node_{h,m}$, that is leaving, deletes it from the key structure. The sibling node $Node_{h,m-1}$ (or $Node_{h,m+1}$, which depends on the tree structure) will be reassigned to the position of its parent node $Node_{h-1,\lceil m/2 \rceil}$. Note, the secret key of the sibling node is unchanged, except the index of the key in the binary tree, that is, $K'_{h-1,\lceil m/2 \rceil} = K_{h,m-1}$.

Step 4 The KDC in the CC updates all $K_{i,j} (0 \le i \le h-2, j = \lceil \frac{m}{2^{h-i}} \rceil)$ with $K'_{i,j} = H(H(K_{i+1,2j-1}), H(i+1, 2j))$.

Step 5 Let $(0 \le i \le h-2, j = \lceil \frac{m}{2^{h-i}} \rceil$, if $2 \times j = \lceil \frac{m}{2^{h-i-1}} \rceil$, and then $l = 2j-1$, else $l = 2j$. Because RCs in-group $GM_{K_{i+1,l}}$ will not know each sibling key of the ancestor keys, they cannot update keys in $AK_{K_{i+1,l}}$, instead they will obtain the keys from the KDC. The KDC encrypts the new keys with $K_{i+1,l}$ and then sends the encrypted message to RCs in $GM_{K_{i+1,l}}$ using the multicast communication protocol.

Step 6 $Node_{h-1,\lceil m/2 \rceil}$ obtains all of the updated keys $K'_{i,j} (0 \le i \le h-2, j = \lceil \frac{m}{2^{h-i}} \rceil)$ from the KDC.

Figure 10 shows an example of the process. In the figure, $Node_{3,8}$ is leaving the system. After an exchange of the *Leave* command and its acknowledgement (Steps 1 and 2 earlier) between the KDC at the CC and $Node_{3,8}$, the procedure is as follows:

Step 3 The KDC as the CC removes $Node_{3,8}$ from the key structure. Then, its sibling node, $Node_{3,7}$, is reassigned to $Node_{2,4}$, with key $K'_{2,4} = K_{3,7}$.

Step 4 The KDC updates $Node_{1,2}$ and $Node_{0,1}$'s keys by $K'_{1,2} = H(H(K_{2,3}), H(K'_{2,4}))$ and $K'_{0,1} = H(H(K_{1,1}), H(K'_{1,2}))$.

Step 5 The KDC encrypts the new key $K'_{0,1}$ of the CC with $K_{1,1}$ and sends the encrypted message $E_{K_{1,1}}(K'_{0,1})$ to those RCs under $GM_{K_{1,1}}$, which are on the CC's left sub-binary tree. Because these RC nodes know $K_{1,1}$, they can decrypt the

DOI: 10.1002/sec

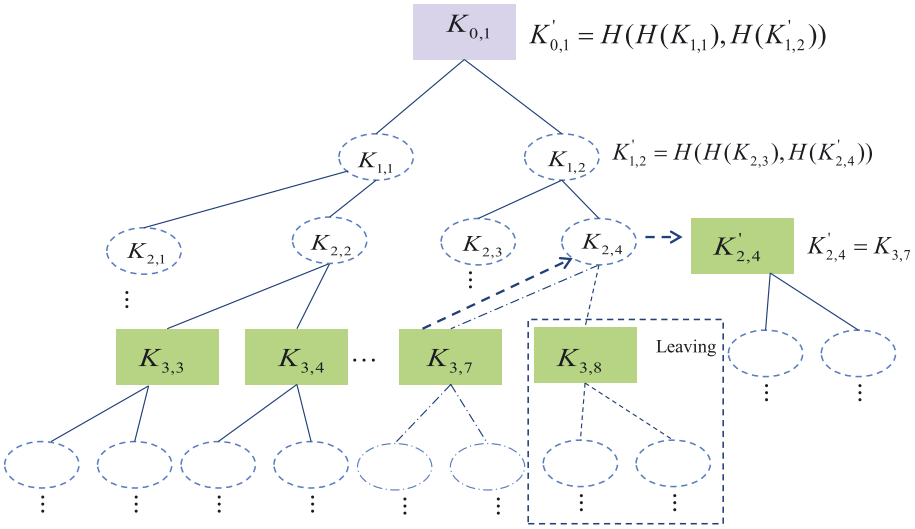

Figure 10. Node leaving phase in the upper level.

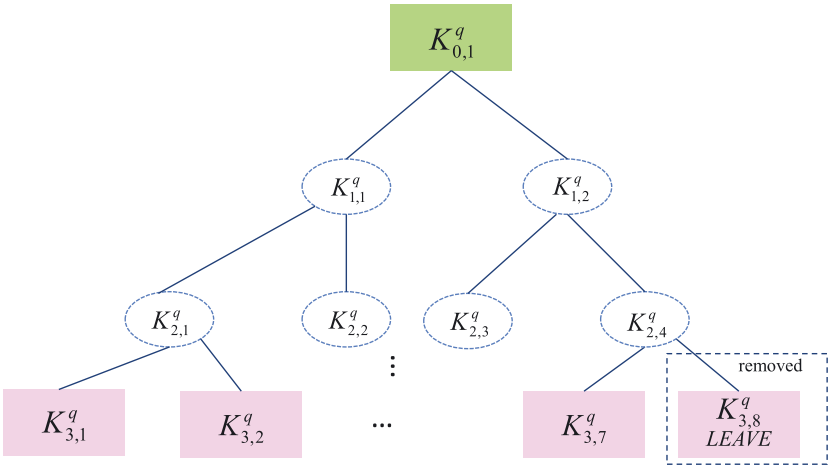

Figure 11. Node leaving phase in the lower level.

message and determine the new key of the root node. Similarly, the KDC encrypts the new keys $K'_{0,1}$ and $K'_{1,2}$ with $K_{2,3}$ and sends the encrypted message $E_{K_{2,3}}(K'_{0,1}, K'_{1,2})$ to those RCs under $GM_{K_{2,3}}$. Because these RC nodes know $K_{2,3}$, they can decrypt the new keys.

Step 6 The KDC computes $E_{K'_{2,4}}(K'_{0,1}, K'_{1,2})$ and sends it to $Node_{2,4}$.

5.2.2. Lower level.

At the lower level, the leaving RE notifies the RC that it would like to leave the system. The key structure will be updated based on the following leave protocol:

(1) The RC broadcasts a leave message $E_{K_{REGION}}(LEAVE, ENTITY_{index})$ to all REs in the region.
(2) All nodes update the new structure of the binary tree and locally update the region key by $K'_{REGION} = H(K_{REGION} \oplus S_{LEAVE})$.

As shown in Figure 11, the leaving node ($Node^q_{3,8}$) first notifies the RC that it is leaving. Then,

(1) RC broadcasts the message $E_{K_{REGION}}(LEAVE, ENTITY_8)$ in the region.
(2) All nodes update the new structure of the binary tree and locally update the region key by $K'_{REGION} = H(K_{REGION} \oplus S_{3,8})$.

5.3. Forward and backward secrecy

As noted in the introduction, the security management architecture should support backward and forward secrecy. Backward secrecy indicates that the future secret keys must inaccessible by former entities. In the upper level, after a node leaves, all the keys on the restricted path from the leaving node are updated; thus, the leaving node cannot do better than a brute force method to compute the new keys.

In the lower level, because the leaving node does not know its restricted secret value, it cannot do better than a brute force method to compute the new keys. Forward secrecy implies that the previous used secret keys must be inaccessible by the new entities. In the upper level, after a new node joins the system, the keys on its restricted path are updated through a one-way hash function, so the new node cannot compute the previous keys. In the lower level, the joining node does not know its restricted secret value, and the region key is updated by hashing the XOR result of the current key with the restricted secret value of the joining node. So the new node cannot determine the previous region key.

6. COMPARISON AND SECURITY ANALYSIS

In this section, we compare our proposed key management schema with the current centralized approach based on the LKH proposed in [12] and the inverse element method proposed in [11]. Both the approaches and our proposed schema support versatile communications, that is, unicast, multicast, and broadcast communications. We compare them based on three categories, which are security and resource utilization, communication costs, and number of keys to store.

6.1. Security and resource utilization comparison

Table V summarizes our comparison based on security and resource utilization of the three schemas. Our proposed schema and Choi *et al*. [12] support a hierarchical structure; however, Kim and Choi [11] do not support it. Our proposed architecture considers the resource differences (i.e., variation in computational resources and physical/electronic security) of entities in the system by splitting the key management into two levels, but both [11] and [12] ignore such differences. All three of the schemas considered use a session key that is generated based on hashing at least a key and a *SID*. Because the hash function is cryptographically secure and the *SID* is independent from previous ones, the session key is updated for each new session. Also, all three schemas compared provide forward and backward security to nodes that join and leave the structure.

Table V. Security and resource comparison.

	[11]	[12]	Our schema
Hierarchical structure	No	Yes	Yes
Resource differences	No	No	Yes
Key freshness	Yes	Yes	Yes
Forward secrecy	Yes	Yes	Yes
Backward secrecy	Yes	Yes	Yes

6.2. Communication costs comparison

In Table VI, we compare the communication cost of the schema proposed here with the key management schemas in [11] and [12]. At the upper level, our schema and Choi *et al*. [12] both use the Iolus framework. Thus, they have comparable overhead for node joining and leaving, which is a function of the number of RC leaves q. The schema of [11] has a flat structure, so it is not comparable. At the lower level, for the node leaving phase of our proposed schema, the RC only sends a broadcast message to all the nodes after receiving the leaving message from the leaving node; thus, the communication cost is $O(1)$, which is the same as that in [11]. However, in [12], the RC sends $\lceil \log_2 r \rceil$ updated keys to the other nodes in the system so the communication cost is $O(\log_2 r)$, where r is the number of REs assigned to a RC. Regarding the node joining phase in the lower level, the RC in the proposed schema sends $\lceil \log_2 r \rceil$ secret values to the new joining node, so the communication costs between the RC and the new joining node are $O(\log_2 r)$. Because the RC only sends one message with the location of the new joining node and the secret value of the new joining node to all the other nodes, the communication cost of the other nodes is $O(1)$. However, the operator in [12] needs to send $\lceil \log_2 r \rceil$ updated keys to the nodes in the systems, so the communication costs is $O(\log_2 r)$. In other words, although the communication cost of node leaving phase on the upper level of the proposed schema is the same as that in [12], the communication costs in the lower level of the proposed schema are greatly reduced, and thereby, the total communication costs are less than [12].

6.3. Storage comparison

Table VII summarized the number of keys stored in the entities at three different schemas. Because the schema in [11] does not support the hierarchical structure and there are only two classes of nodes in this schema, so, it is hard to compare it with the proposed schema, and we mainly compare with the schema in [12]. In the table, q denotes the number of RCs in the system, and r denotes the number of REs attached to a RC. Regarding to the number of keys stored in a CC or a RE, there is no difference between our proposed schema and the schema in [12]. However, Figure 12 shows the comparison of the number of keys stored in a RC according to the number of RCs and the number of REs under this RC between the two schemas. We can see that in the proposed schema, a RC stores fewer keys than that in [12].

As an illustration in a large power grid, we consider the Los Angeles Department of Water and Power (LADWP) system. The LADWP [28] is the largest public owned power company in the USA operating a system consisting of one energy control center and 196 substations and serving 1 461 344 customers. The LADWP grid has over 14 000 miles of transmission and distribution power lines, and plans are in place to deploy smart meters to

DOI: 10.1002/sec

Table VI. Communication cost comparison of key management schemas.

		[11]	[12]	Our schema
Cost of Leaving in upper level	CC	*N/A*	$O(\log_2(q))$	$O(\log_2(q))$
	Leaving node		$O(1)$	$O(1)$
	Others		$O(\log_2(q))$	$O(\log_2(q))$
Cost of Joining in upper level	CC	*N/A*	$O(\log_2(q))$	$O(\log_2(q))$
	Joining node			
	Others			
Cost of Leaving in lower level	RC	$O(1)$	$O(\log_2(r))$	$O(1)$
	Leaving node		$O(1)$	
	Others		$O(\log_2(r))$	
Cost of Joining in lower level	RC	$O(\log_2(r))$	$O(\log_2(r))$	$O(\log_2(r))$
	Joining node			
	Others	$O(1)$		$O(1)$

CC, control center; RE, remote end; RC, regional coordinator.

Table VII. Number of keys to store comparison.

		[11]	[12]	Our schema
Num of keys to store	A CC	*N/A*	q	q
	A RC	2	$1+r+\log_2(q)$	$2+\log_2(q)$
	A RE	$1+\log_2(r)$	$1+\log_2(r)$	$1+\log_2(r)$

CC, control center; RE, remote end; RC, regional coordinator.

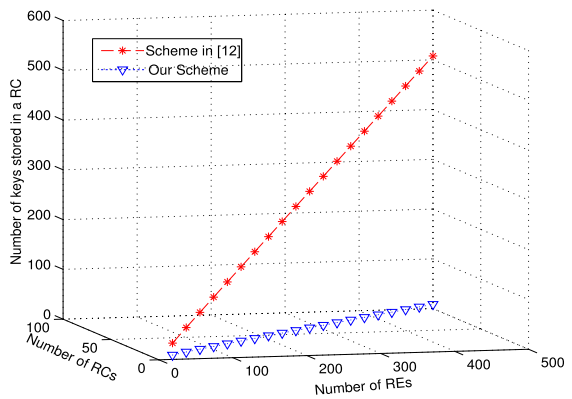

Figure 12. Number of stored keys in a regional coordinator (RC) versus number of remote ends (REs) under RC and number of neighbor RCs.

customers along with 10 000 micro-remote terminal units (micro-RTUs) for monitoring power quality and fault diagnosis. Assuming 1 meter per customer and that the customers are equally distributed among the 196 RCs, then there are 1 471 344 REs (including the micro-RTUs) resulting in 7507 REs per RC. According to Table VII at the lower level, in our schema, each RC only needs to store only 10 keys. However, each RC needs to store 7518 keys if using the schema in [12].

In a summary, on the aspect of security, our proposed schema could achieve the same lever of security guarantee as the other two approaches—providing both forward secrecy and backward secrecy. Our proposed schema and the schema proposed in [12] support the hierarchical structure for the smart grid systems, but the schema proposed in [11] is not able to supports that structure. On the aspect of the communication costs and storage requirement, our proposed schema is more efficient in communication and utilizes less storage compared with the schema proposed in [12].

7. CONCLUSIONS

In this paper, we propose a key management architecture to enable secure encrypted data communications in smart grid systems. The architecture is multi-level in structure, which provides several advantages, namely, ease in allowing for differences in computational resources among grid components, ability to tolerate the differences in electronic/physical security between core and edge grid devices, and flexibility in adapting to components joining and leaving the power grid. Also, the key management architecture is arranged to fit the hierarchical control structure of the smart grid. The architecture provides for versatile secure communications, supporting encrypted unicast, multicast, and broadcast communications with low communication overhead. Furthermore, the

architecture provides forward and backward secrecy in the event nodes join or leave the grid. Comparing the proposed schema with existing ones, the proposed schema requires less communication overhead and storage costs, and it supports the hierarchical structure of the smart grid control system.

REFERENCES

1. Yan Y, Qian Y, Sharif H, Tipper D. A survey on smart grid communication infrastructures: motivations, requirements and challenges. *IEEE Communications Surveys & Tutorials* 1st Quarter 2013; **15**(1): 5–20.
2. Yan Y, Qian Y, Sharif H, Tipper D. A survey on cyber security for smart grid communications. *IEEE Communications Surveys & Tutorials* 4th Quarter 2012; **14**(4): 998–1010.
3. Fang X, Misra S, Xue G, Yang D. Smart grid the new and improved power grid: a survey. *IEEE Communications Surveys & Tutorials* 4th Quarter 2012; **14**(4): 944–980.
4. Long X, Tipper D, Qian Y. An advanced key management scheme for secure smart grid communications. *2013 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Vancouver, Canada, 2013; 504–509.
5. Bakken D, Bose A, Hauser C, Whitehead D, Zweigle G. Smart generation and transmission with coherent, real-time data. *Proceedings of the IEEE* 2011; **99**(6): 928–951.
6. Taft J, Martini PD. *Ultra large scale power system control architecture*. Available from: http://www.gridwiseac.org/pdfs/cisco_control_architecture_white_paper.pdf [Accessed on July 5, 2016].
7. Taft J, Martini PD. Ultra-large scale control architecture. *Proceedings of IEEE Innovative Smart Grid Technologies (ISGT)*, Washington, DC, 2013; 978–983.
8. *NERC. Cip cyber security standards.* Available from: http://www.nerc.com/pa/stand/Pages/default.aspx [Accessed on July 5, 2016].
9. Fries S, Hof H J. Smart grid security standardization. In *Smart Grid: Applications, Communications and Security*, Berger L, Iniewski K (eds). John Wiley, 2012.
10. Mittra S. Iolus: a framework for scalable secure multicasting. *Proc. of ACM SIGCOMM '97*, New York, NY, 1997; 277–288.
11. Kim J, Choi H. An efficient and versatile key management protocol for secure smart grid communications. *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC) '12*, Shanghai, China, 2012; 1823–1828.
12. Choi D, Lee S, Won D, Kim S. Efficient secure group communications for scada. *IEEE Transactions on Power Delivery* 2010; **25**(2): 714–722.
13. Jøang A, Pope S. Semantic constraints for trust transitivity. *Proceedings of 2nd Asia-Pacific Conference Conceptual Model (APCCM) '05*, Darlinghurst, Australia, 2005; 59–68.
14. Jsang A, Ismail R, Boyd C. A survey of trust and reputation systems for online service provision. *Decision Support Systems* 2005; **43**(2): 618–644.
15. Liang J, Naoumov N, Ross KW. The index poisoning attackin p2p file sharing systems. *Proceedings of IEEE IEEEINFOCOM '06*, Barcelona, Spain, 2006; 1–12.
16. Bobba RB, Dagle J, Heine E. Enhancing grid measurements: wide area measurement systems, naspinet, and security. *IEEE Power and Energy Magazine* 2012; **10**(1): 67–73.
17. Wong C, Gouda M, Lam S. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking (TON)* 2000; **8**(1): 16–30.
18. Howarth MP, Iyengar S, Sun Z, Cruickshank H. Dynamics of key management in secure satellite multicast. *IEEE Journal on Selected Areas in Communications* 2004; **22**(2): 308–319.
19. Ren K, Zeng K, Lou W. A new approach for random key predistribution in large-scale wireless sensor networks. *Wireless Communications and Mobile Computing* 2006; **6**(3): 307–318.
20. Ren K, Lou W, Zeng K, Moran P. On broadcast authentication in wireless sensor networks. *IEEE Transactions on Wireless Communications* 2007; **6**(11): 4136–4144.
21. Law YW, Palaniswami M, Kounga G, Lo A. Wake: Key management scheme for wide-area measurement systems in smart grid. *IEEE Communications Magazine* 2013; **51**(1): 34–41.
22. Nicanfar H, Jokar P, Beznosov K, Leung VCM. Efficient authentication and key management mechanisms for smart grid communications. *IEEE Systems Journal* 2014; **8**(2): 629–640.
23. Baugher M, Canetti R, Dondeti L, Lindholm F. Multicast security (MSEC) group key management architecture. *RFC 4046* 2005.
24. Sun Y, Trappe W, Ray KJ. A scalable multicast key management scheme for heterogeneous wireless networks. *IEEE/ACM Transactions on Networking (TON)* 2004; **12**(4): 653–666.
25. Kong J, Lee YZ, Gerla M. Distributed multicast group security architecture for mobile ad hoc networks. *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC) '06*, Las Vegas, NV, 2006; 640–645.

26. Cao H, Zhu P, Lu X, Gurtov A. A layered encryption mechanism for networked critical infrastructures. *IEEE Network* 2013; **January/February**: 12–18.
27. DoE US. United states department of energy communications requirements of smart grid technologies. Available from http://energy.gov/gc/downloads/communications-requirements-smart-grid-technologies [Accessed on July 5, 2016].
28. Los angeles department of water and power. Available from http://www.ladwp.com/ [Accessed on July 5, 2016].