Lightweight Key Management Protocols for Smart Grids

Yongdong Wu, Vivy Suhendra, Hendra Saputra, Zhigang Zhao
Institute for Infocomm Research, Singapore
{wydong,vsuhendra,hsaputra,zzhao}@i2r.a-star.edu.sg

Feng Bao
Shield Lab, Huawei International Pte. Ltd
bao.feng@huawei.com

***Abstract*—Smart grid is a critical power infrastructure and its security is an essential requirement. This paper presents security protocols for securing its communication, control and management. Specifically, it presents a key agreement and update protocols based on symmetric-key primitives and smart grid topologies. The paper enables to efficiently detect and revoke compromised IEDs.**

***Index Terms*—Key management, smart grid, SCADA**

I. Introduction

Since Thomas Edison built the first complete electric power system in 1882, the power system has been the largest machine and played a critical role in the world. In the past, the core of the system is mostly connected using dedicated networks. As such, its security is largely achieved through the closed nature of the physical networks. Recently, with the advance of information and communication technologies, the power grid has emerged as a modernized smart grid for improved control, efficiency, reliability, and safety, and hence attracted great interest all over the world. The International Energy Agency projected that cost-effective smart grid deployment is essential to meet the global need of clean energy in future economic development and climate change mitigation [1]. However, the emerging smart grid, while benefiting the legitimate participants (i.e., electric utilities, consumers, service providers), also presents unprecedented opportunities for adversaries [2]. Even worse, the complexity of the grid makes it difficult to fully secure, as evidenced by continuing discovery of vulnerabilities in the grid devices, even from leading smart grid solution providers [3].

Smart grid consists of many devices such as the Programmable Logic Controller (PLC), Remote Terminal Unit (RTU), status sensor, and smart meter. Typically, a Supervisory Control and Data Acquisition (SCADA) system is used to control the devices of smart grids by acquiring their status information. However, devices at the grid periphery, such as RTUs and smart meters, are especially susceptible to attacks on their status and control data as they are often physically accessible to the adversary [4], and these vulnerabilities provide new entry points to compromising the whole grid. In other words, smart grid faces a multitude of attacks coming from the cyber world and the physical world [5]–[8]. The cyber attacks may lead to eavesdropping of private information or cause misbehaviors of physical components managed by software routines, while the physical attacks may damage the smart grid infrastructure or facilitate the launch of cyber attacks. As the Electric Power Research Institute advises: "*Vulnerabilities might allow an attacker to penetrate a network, gain access to control software, and alter load conditions to destabilize the grid in unpredictable ways*" [9]. These attacks may manipulate the data flow in the network, compromise software/hardware of grid devices, or overwhelm the communication and computational resources to cause delay or failure of communication.

Such impending threats will become more severe when emerging trends in information and communication technologies for smart grids are deployed to enable real-time data collection and automatic control without protection or with mis-configured protection. By tampering with insecure links, an adversary is able to intercept, alter or forge data [10]–[12] and hence cause damages in various aspects such as: (1) the infringement of customers' privacy, where the intercepted information apparently reveal whether a particular customer is at home at a certain time based on the fingerprint of power usage [13]; (2) the rise of energy theft incidents in places such as Ireland, Hong Kong, and Virginia, where customers attempted to illegally lower their electricity bills via meter tampering, bypassing, or other unlawful schemes [14]—notably, energy theft is estimated to cost the energy industry approximately $6 billion each year in USA [15]; (3) inadmissible system state which could lead to disasters of significant social and economic consequences; (4) more ways to compromise smart grid devices with expected new functionalities such as remote relays in distribution automation or remote disconnect options. Once the devices are compromised, the attacker will be able to disrupt normal power grid operations, causing blackout and other devastating effects on critical infrastructures in the physical world [16]. To prevent such damages, security has been a major emphasis in the design of SCADA system of smart grids, especially after the Stuxnet incident in 2010 [17].

Well-known solutions for protecting SCADA data transmission between devices are based on asymmetric key cryptography or Public Key Infrastructure (PKI) [18]. For instance, the international standard IEC 62351 [19] focuses on PKI for data and communications security in smart grids. Such schemes incur tedious digital certificate management, including massive certificate deployment, renewal, and revocation, on top of the high demand on device computational resources.

To provide light-weight security for resource-limited field devices, symmetric key schemes are designed to secure the communication between two or more devices via shared secret keys. As a smart grid usually has a huge number of field devices, it will likewise have to manage a huge number of keys. Therefore, a major challenge of the symmetric key scheme is how to manage the secret keys efficiently. Existing schemes generally differentiate keys for unicast/multicast/broadcast communications, and hence various key architectures are studied in [20] in terms of key storage requirement and support for unicast/multicast scenarios. A naïve solution is to assign the same key for all the devices in the system to achieve the simplest key management. However, this single-key solution has very low security robustness as compromising one device means compromising the entire system [21]. A better solution is to adopt a two-level system, where a central server (e.g., Substation Automation System or SAS for short) shares one *group key* with all field devices for multicast/broadcast messages, and one *pairwise key* with each field device for unicast messages. Clearly, as the number of keys stored in the server is linear to the number of field devices, this solution is lack of scalability. Recent solutions have focused on designing the key structure for efficient management, e.g., organized in a key graph [22] or a binary tree [23]. However, complex key structures also incur maintenance overhead when a new device is added to the graph/tree or an old device is removed from the graph/tree.

This paper presents a novel key management scheme for smart grid devices based on symmetric key primitives and smart grid topologies. Specifically, the scheme has three components: (1) the *key initialization* protocol to authentically install a key into a new device in an insecure remote location by exploiting proximity and audiovisual signals [24]; (2) the *session key generation* protocol for field devices distributed over wide areas with the key installed by the first component; (3) the *session key update* protocol to provide field devices with persistent security over tens of years, as well as enable detection of compromised devices. Analysis of the scheme shows that it is efficient and applicable to smart grids, and the implementation of all these components demonstrate the soundness of the proposed scheme.

The remainder of this paper is organized as follows. Section II elaborates the components in the proposed scheme. Section III analyzes the proposed components in terms of security and performance. Section IV introduces the implementation and simulation results. Finally, Section V draws a conclusion.

II. Proposed Scheme

A. System model

With regard to Fig. 1, a smart grid is managed by a complicated SCADA system [25]–[27], consisting of control center, SAS, and field devices such as PLCs, RTUs, and smart meter. The SAS may further comprise various types of equipments including network device, user interface, firewall, Intelligent Electronic Device (IED), remote access point, Data Aggregator (DA), and Key Distribution Center (KDC). The SAS/PLC/RTU can be regarded as the "brain" of the SCADA system as the actual control program for a given process or its control systems is executed within them. They can work with either local or remote inputs and outputs.

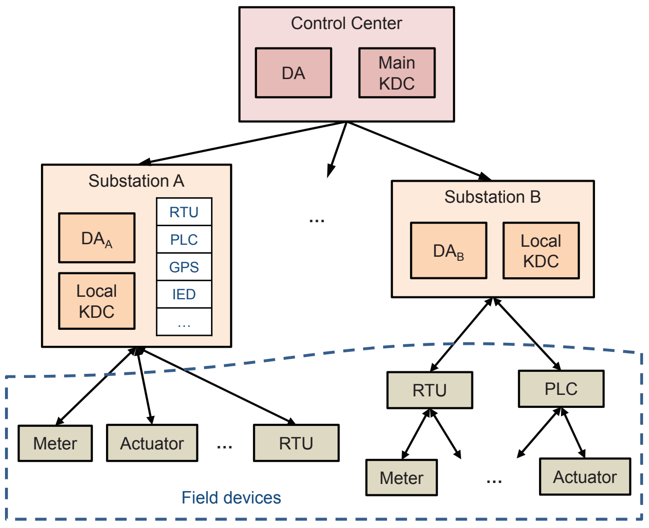

Fig. 1. The architecture of a SCADA system. DA and KDC may be included in the control center and/or SAS.

All devices and subsystems in the SCADA system are connected with networks. If the networks are not protected, it is possible for attackers to manipulate grid data such as smart meter measurements or control data [10]–[12]. To enable secure communications, the devices must be initialized with keys, which can be either public keys or secret keys depending on the cryptographic suite in the devices. As many resource-constrained IEDs in smart grids are evolved from traditional power grids and not trivial to replace, the scheme presented in this paper aims to make good security affordable by those low-end devices. At the same time, the scheme also strives to optimize computation and storage required from the KDC, in view of the potential scale of the grid. Thus, symmetric key primitives are employed in the presented scheme unless otherwise stated.

B. Session key structure

The presented scheme uses distinct *session keys* to protect individual communication channels between any pair of devices. The session keys can be considered a hybrid between long-term keys and one-off keys used to protect single communication instance, in that they are valid for a strictly limited time period, such as a day or a week. Within the validity period, they are used directly as authentication and encryption keys. This arrangement achieves two features: the limited validity period ensures that any compromise is short-lived, while the direct usage eliminates the overhead for key establishment steps preceding each communication instance.

In the proposed key structure, each device uses one session key for each distinct communication channel. Here, *channel* refers to a combination of two factors: communication counterpart (sender/receiver) and communication mode (unicast, multicast, or broadcast). In other words, each session key is a secret shared with exactly one other counterpart, either another device or a group of devices. Fig. 2 shows an example, where Substation A shares a broadcast session key BK_A with all field devices it manages, and unicast session keys SK_{Aj} privately with each field device Aj; and it shares with the Control Center a broadcast session key BK and a unicast session key SK_A. As the number of broadcast/multicast keys is much smaller than the number of unicast keys, the management of broadcast/multicast keys is relatively easier. For simplicity, we focus on the management of unicast keys in the following.

As the same figure illustrates, session keys are intended to cover only one level of the management hierarchy. That is, a device shares session keys with its direct manager and its direct subordinates only. This keeps the number of keys at field devices manageable. Session keys are generated and managed by a KDC. In addition to the main KDC at the grid control center, local KDCs are placed at sub-management centers. These can be substations, sub-MTUs (Master Terminal Units), and other subsystems that have devices under their management. This distributed key management also serves to reduce reliance on one centralized server.

We note that the hierarchical segmentation is a typical arrangement in smart grids. Nevertheless, if future use cases emerge where the Control Center needs to communicate directly to a field device, they can be achieved in the hop-by-hop way: from Control Center to the managing substation and then from the managing substation to the field device, with the communication at each step protected with the respective session keys of the parties involved. If end-to-end security is desired, the main KDC of the Control Center and the field device shall set up their own shared key in advance.

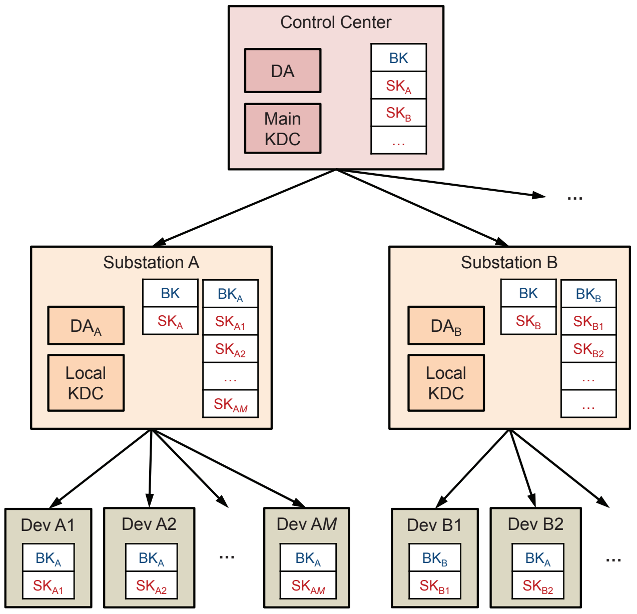

Fig. 2. The session key structure. The Control Center and substations may each maintain a KDC, a DA, and a key table.

C. Session key management

To maintain long-term security, every session key shall be valid only for a limited time period. Upon expiry of a session key, the KDC (Main or Local) initiates the key update protocol for the affected device to update its key. It is recommended to choose irregular key update timings within the acceptable range (e.g., a randomly chosen time during off-peak hours every 6-7 days), to make the process unpredictable for attackers to observe. One possible implementation is for the KDC to store expiry times along with the session keys in its Key Table. When a session key is newly created or updated in the Key Table, its expiry time is set to a new non-deterministically generated value.

In the rest of the discussion, the session key shared between the KDC and a device j during the i-th period is denoted as SK_j^i, or simply SK^i when focusing on a single device. The session keys of a device across update periods are related in a dynamic key series $SK^1, SK^2, ..., SK^n$, where

$$SK^i = \mathcal{H}(SK^{i+1}), \quad i = 1, 2, ..., n-1 \tag{1}$$

$\mathcal{H}(\cdot)$ is a cryptographic hash function as before, and the number of update periods n can be chosen according to the desired key update frequency over the system lifetime.

The following describes the first session key setup protocol for a field device FD within a management domain, which may be carried out immediately after FD is installed on the field.

(1) The managing KDC chooses a unique value for SK^n, derived from its own secret SK, a nonce such as a timestamp, and other parameters such as the device ID (e.g., $SK^n = \mathcal{H}(SK||TimeStamp||DevID)$). It then computes the dynamic key series as per Eq. (1) to obtain SK^1.
(2) The KDC sends a key setup instruction along with the key SK^1 to FD. The message is secured with the pre-installed key.
(3) Upon receiving and deciphering the message, FD stores SK^1 as its session key. It then sends a confirmation message to the KDC, secured with the newly set session key SK^1.
(4) Upon receiving and deciphering the confirmation, the KDC adds a Key Table entry for FD, where it stores the current session key SK^1, the current period $i = 1$, key input values (nonce, etc.), and sets an expiry time. All intermediate computation including SK^n is deleted.

The above key setup protocol is repeated for each session key assigned to the field device upon initialization (e.g., once for the unicast session key and once more for the broadcast session key). For broadcast session key setup, the KDC performs step (1) to generate the first broadcast session key for the managed group of devices only once in the beginning. With every field device added afterwards, the active key is retrieved from the Key Table to send to the new device.

Subsequently, the KDC routinely scans the Key Table for expiring keys. Once expiry of the i-th session key for a field device FD is met, the key update protocol proceeds as follows.

(1) The KDC recomputes SK^n from the stored key SK and performs $n-i-1$ round(s) of the key series computation to reach SK^{i+1}.

(2) The KDC sends a key update instruction along with the new session key SK^{i+1} to FD. The message is secured with FD's current session key SK^i, extracted from the Key Table.

(3) Upon receiving and deciphering the message, FD verifies the received key $\widetilde{SK}^{i+1}$ by checking whether $\mathcal{H}(\widetilde{SK}^{i+1})$ matches the current key SK^i. If this verification fails, FD ignores the message.

(4) If the verification succeeds, FD overwrites its stored session key with the new key $SK^{i+1} = \widetilde{SK}^{i+1}$. It then sends a confirmation message to the KDC, secured with the new session key SK^{i+1}.

(5) Upon receiving and deciphering the confirmation, the KDC updates its Key Table entry for FD, overwriting the current session key with SK^{i+1}, increasing the update period $i \leftarrow i+1$, and setting a new expiry time. All intermediate computation including SK^n is deleted.

(6) If the wait for device confirmation times out without any message received, the KDC may re-send the key update instruction, up to a pre-determined retry limit.

In the broadcast case, the KDC sends the new broadcast key to all affected field devices in step (2), and each device performs the confirmation step independently. The KDC may opt to wait for confirmation messages from all devices, or to require no confirmation message and simply update its Key Table immediately.

III. Analysis

A. Security

1) Novel key management scheme for specified security features: The dynamic key management scheme governs the encryption and authentication of the system. In our scheme, the secret keys are updated periodically and are self-verifiable. Its security strength is higher than that of the traditional security, in the sense that temporary compromise of a field device will not cause any damage in the next period unless the adversary is able to continuously compromise the device.

2) Secure communication between devices: The dynamic key management scheme is efficient in the sense that no PKI operation is required in any key update process. The design objective is efficiency for any two communicating entities.

3) End-to-end security: The security design aims at end-to-end security to enable appropriate response to incidents such as impersonation of a particular user or meter and theft / illegal modification of meter credits. Data from field devices such as smart meters should be protected all the way from the device to its (local) management center. Therefore, the secret key is shared by the center and the particular device only.

B. Session key management

1) Properties of session key update: The i-th session key update for a field device replaces its current key SK^i with the new key SK^{i+1} according to Eq. (1). Therefore,

- SK^i can be efficiently calculated from SK^{i+1}, enabling the field device receiving the new key SK^{i+1} to perform the calculation and check the result against its current stored key SK^i;
- It is computationally hard to calculate SK^{i+1} from SK^i, preventing an attacker who has discovered the current key SK^i from obtaining the new key SK^{i+1} and continuing to compromise the communication.

The dynamic key series is similar in principle to the hash chain [28] that underlies several key evolution mechanisms proposed for wireless sensor networks. In particular, [29] combined a hash chain and a reverse hash chain to provide both future and past key secrecy, but only for group keys; pairwise keys are chosen randomly. The scheme requires sensor nodes to store two current key ingredients (from both chains) and the key counter in addition to the key itself, to enable future key computation and version verification / synchronization. While past key secrecy is useful to protect the privacy of past data transmissions in the smart grid, it is less critical in preventing consequential attacks on the grid that require real-time communication tampering using current or future keys. As such, this paper does not offer past key secrecy in favor of deployability to low-end field devices. In all, the key update method has the following properties within the capability of field devices.

- *Self-verifiability:* The field device is able to verify that the new key it receives is an authentic key generated by the KDC before accepting and using the key. If an attacker somehow manages to capture an old key update packet and re-forward it to the device, or to impersonate the KDC in sending his own chosen key, the device is able to recognize the fake update and reject it.
- *Future key secrecy*: If a session key that has been discovered by an attacker is subsequently updated, the attacker cannot simply infer the new key from the key he has, thus unable to compromise the channel for any extended length of time.
- *Low cost on device:* The key management incurs minor computational and storage cost on the device's end, and is thus deployable on any low-resource field device.

2) Rekeying requirement: When a new field device joins a managed domain, the domain KDC performs key initialization followed by key setup protocols for all session keys assigned to the device. Unicast session keys are freshly generated specifically for the new device; while for broadcast channels, the new device is simply informed of the active broadcast session keys for every group that it is added to. There is no rekeying required for existing devices in the same group.

When a field device is taken off a managed domain, the domain KDC removes the device from all group lists and removes the device's entry from its Key Table. The KDC then initiates an update of broadcast session keys for each of the device's former groups. As such, rekeying upon device removal simply involves a routine key update process.

C. Compromised device detection

The secret key series of the key update scheme have one-way feature, that is, it is computationally hard for anyone but the KDC to generate SK^{i+1} just by knowing SK^i or earlier keys. On the other hand, it is self-verifiable in the sense that upon receiving SK^{i+1}, the device itself can immediately and easily verify if it is a correct update of SK^i. In other words, the update is not random, but verifiable. The time to update SK^i to SK^{i+1} is determined by the KDC, which is irregular and unpredictable by other parties. Therefore, if the key update protocol fails to complete, the KDC deduces the reasons for failure as

- The field device has been tampered with and its stored session key changed or corrupted, causing deciphering failure at the device's end; or
- The message containing the new session key has been intercepted and/or modified, causing message loss, deciphering failure, or key corruption leading to verification failure at the device's end; or
- The message containing the confirmation has been intercepted and/or modified, causing message loss or deciphering failure at the KDC's end.

As such, consecutive failures of the key update protocol will alert the KDC that the field device or the communication channel may have been compromised. It may then take action such as sending an engineer to reset the device.

D. Length of key series

The KDC and each field device share a secret key SK^1 at the beginning. The secret key is updated dynamically, that is, $SK^1 \rightarrow SK^2 \rightarrow ... \rightarrow SK^n$. The choice of the value n is a balance of factors including desired update frequency (e.g., higher for less secure environment), ease of reconfiguration when a key series is exhausted, and performance overhead for updates considering other operational taskload and the number of managed devices, among others. To illustrate, a system lifetime of 50 years with averaged weekly key updates can be supported with $n = 50 \times 52 = 2,600$, while averaged daily key updates calls for $n = 50 \times 365.25 \approx 18,263$. When n runs out, the field device should be re-initialized to accept a new key series.

E. Computational and storage demand

1) Server: For each session key, the update at the end of the i-th period requires the KDC to re-generate SK^n and perform $n - i - 1$ round(s) of the chosen one-way function, with i running from 1 to $n - 1$. While KDCs are typically high-performing systems capable of handling this, the overhead may be significant if n is large, on top of the huge number of managed devices. This can be alleviated by partial storage of the key series, where the KDC stores the values at pre-determined intervals of the key series (e.g., $SK^{100}, SK^{200}, ...$), and starts the computation of SK^i from the nearest future key (e.g., SK^{168} computation will start from SK^{200}). Clearly, SK^i can be purged from the partial storage after the i-th update. This optimization will limit computational per update to at most $w - 1$ rounds for a chosen interval of w, at the expense of extra server storage for $\lceil n/w \rceil$ key values per device.

2) Field device: A field device keeps one key value per communication channel. Given that the grid typically has a hierarchical operational structure and very specific role assignment to field devices such as smart meters, each device should only need to communicate with a small set of other grid devices (e.g., control center, substation), thus limiting the number of channels to a manageable number.

In each key update protocol, the device performs one computation of the cryptographic hash function $\mathcal{H}$ to verify the received key. As the device should already be equipped with an encryption function to secure its messages, and possibly also a hash function for checksum or Message Authentication Code (MAC), those routines can be reused for $\mathcal{H}$, thus incurring no additional code storage.

3) Network traffic: Each key update protocol involves an exchange of only 2 messages, as the protocol is self-verifiable. The confirmation message may even be made optional if flooding the KDC is a concern when updating a large number of devices. The resulting single-message protocol would however lose the compromise detection capability.

F. Comparison with existing schemes

Table I shows the features of our scheme alongside several comparable key management schemes commonly cited in the smart grid literature. For comparison across the different proposals, the terms for smart grid entities have been unified to *CC* for the control or operation center, *Substn* or *Manager* for substation or regional center or data collector, and *FD* for field devices or smart meters. This management hierarchy is compatible across the schemes, although different communication needs are considered, as reflected in the managed channels. Long *et al.* [33], Uludag *et al.* [34], and this paper account for the role of the central server, while the rest consider only localized communication following typical grid operations. While Uludag *et al.* provide for CC-FD end-to-end security at the cost of managing extra secret keys, Long *et al.* simply route CC-FD communication through the substation.

Our scheme also uses the per-channel, short-lived session key directly for securing communications for minimum overhead, while all other schemes derive single-use keys[1] from the node key or the channel key in the case of Uludag's. Notably, many schemes do not consider periodical key update, and updates only group keys upon device addition or removal from the group. The schemes that do (hash-update [30], [31]) use hash chain, deriving the new key by applying hash function on the current key in order to provide key independence only. In contrast, our scheme uses reverse hash chain to also prevent any key compromise to lead to inferrence of future keys (future key secrecy). Our unified treatment of group keys as per-channel keys also simplifies the rekeying effort when a device

[1]Note that some papers call this "session key", with "session" referring to a communication instance, while in our context it refers to a relatively short time interval.

TABLE I
SCHEME COMPARISON

Scheme	**[30] [31]**	**[32]**	**[33]**	**[34]**	**This paper**
Managed channels *uc:* unicast; *mc:* multicast incl. broadcast	FD-Manager uc/mc	FD-Manager uc/mc	- CC-Substn uc/mc - Substn-Substn uc/mc - Substn-FD uc/mc - FD-FD uc (same Substn)	- CC-Substn uc - CC-FD uc - Substn-FD uc - CC-FD mc via Substn	- CC-Substn uc/mc - Substn-FD uc/mc - CC-FD uc/mc (optional)
Key structure	Key graph, but unrelated node keys	One-way function tree (OFT)	*CC-Substn:* Binary tree with Iolus framework *Substn-FD:* Inverse Element	Per-channel shared secret	Per-channel shared secret
Node key	Centralized generation	Identity-based public/private key	Secret key / key set and region key	Public/private key with centralized generation	N.A. (channel perspective)
Communication key	Single-use key				Channel key
Key renewal scheme	Hash chain	–	–	–	Reverse hash chain
Key renewal protocol	Independent renewal by each party	–	–	–	Transmission from KDC with FD-side verification
Rekeying on device addition / removal	New key generated, unicast to each node in updated group	Tree structure update, affected key update, multicast notification	Tree structure update, affected key update and propagation	–	*Addition:* Key setup for new device only *Removal:* Trigger routine key update
Properties	Forward and backward secrecy	- Forward secrecy - Group key secrecy - Collusion-resistant past and future group key secrecy - Limited impersonation to single identity	Past and future group key secrecy	- Remote initialization - Replay attack detection	- Remote initialization - Self-verifiable key update with compromise detection - Future key secrecy - Replay attack detection

joins or leaves a group, compared to key tree-based schemes such as SKM and Long *et al.*

The last row of the table summarizes the properties of each scheme. As schemes other than ours use single-session keys whose derivation involve a nonce component, they are able to provide forward and/or backward secrecy, in the sense that a compromise of a node/channel key does not disclose older or newer communications. It does not necessarily prevent impersonation that might lead to discovery of future keys. For schemes with only group key update mechanism, the concept of future or past key secrecy refers to before and after group member changes (device addition or removal). Thus, in the absence of group member changes, a compromise of a node/channel key may go undetected and uncorrected.

IV. IMPLEMENTATION

A. System configuration

The system setup for confidentiality and communication security tests comprises a substation and a field device. The substation has a KDC and a DA, both implemented as Java applications on JRE 1.6 with JCE support. The KDC is equipped with session key generation and exchange/update protocols, with the AES block encryption function as $\mathcal{H}$. The key series length n is set to 5000, with key storage interval at 500. All session keys are 128 bits long. The DA is equipped with message packing/unpacking routines which include encryption/decryption and MAC. The encryption/decryption algorithm used is AES in Cipher Block Chaining (CBC) mode [35], with 128-bit key size and PKCS#5 padding [36]. The MAC algorithm used is CBC-MAC [37] using AES as the block cipher, built on the same encryption function implementation. The first 96 bits of the CBC-MAC(AES) 128-bit output is used as MAC.

The field device used is the STM32F100RB processor (16 Kbyte FLASH, 4 Kbyte RAM, 8 MHz ARM CPU clock). It is equipped with the same message packing/unpacking algorithms as the DA, implemented in C. To account for real-time tasks of the device such as sampling the power consumption every 4 ms, the time requirement is set to less than 0.5 ms for one AES encryption operation. To process large-sized data that exceed RAM capacity, the algorithm reads one 16-byte block of data from FLASH to RAM each time, performs block encryption/decryption, and then sends the processed block out before fetching the next block.

Two communication channels are configured in the experiment: DA$\leftrightarrow$device two-way unicast and DA$\rightarrow$device broadcast. Thus the substation and the field device each maintains two session keys for the two channels. Messages are tagged with an 8-bit value indicating the key used (unicast or broadcast). In this simple case, only the field device needs to check the key tag to decipher received messages, as the DA receives only unicast messages from the field device.

B. Firmware upgrade

The support for remote firmware upgrade for field devices, which has high security risk and involves large-sized data transmission, is achieved in two steps. Firstly, the KDC supplies the DA with a random one-time key OTK, which is used to encrypt the firmware code. The resulting ciphertext is packed as a broadcast message (i.e., secured with the corresponding broadcast session key) and sent out to all affected field devices. Lastly, the DA sends the key OTK individually to each field device, secured with their respective unicast session key. The combination of broadcast and unicast transmissions achieves unicast-level security while avoiding linear scaling of the security overhead.

C. Replay attack detection

To detect replay attacks, a 32-bit sent-message counter is included in each packet exchanged between the field device and the DA. A separate counter is maintained for each communication channel shared with each counterpart, similar to session keys. The counter indicates the number of messages that the sender (DA or field device) has sent through the particular channel, and is incremented at the sender's end. The recipient of the message can compare the counter value in the message with the counter value it has stored from the last received message to determine if a replay attack has occurred. In general, the received counter is not expected to be a strict increment-by-one from the last received value, as there may be admissible cases of message loss (e.g., a field device missing broadcast messages while temporarily down for maintenance).

It is possible for the counter to overflow[2], that is, when its value reaches $2^{32} - 1$, the next increment will reset it to 0. Such an incident will invalidate the replay attack detection that depends on a simple comparison of message counter values. To overcome this issue, the improved detection mechanism makes use of the regularly updated session key in addition to the message counter. Essentially, it makes sure that an overflow will not occur *within* a single key update period, and hence sent messages can be counted relative to the start of the key update period. An auxiliary *anchor counter* at the recipient's end stores the counter value received from the latest successful key update, as the basis to adjust counter values of subsequently received messages, modulo 2^{32}. This method relies on two assumptions: (1) at most 1 message is sent per millisecond; (2) the session key is updated at least monthly, which means the maximum length of one period is approximately 2.68×10^9 ms, allowing $2.68 \times 10^9 < 2^{32}$ sent messages. Based on the first assumption, the counter routine is constrained to accept an increment at most once in a millisecond.

An additional measure is taken in the boundary case whereby the received counter is equal to the stored last counter (both relative to the anchor). To distinguish a legitimate case from a replay attack, recipients partially store the MAC from recent messages to compare with the MAC of the newly received message. In this experiment, the last 4 bytes of MAC from 10 latest messages are stored. If the new message MAC can be found in recent history, it is a replay attack; otherwise, it is a legitimate case. The full method implementation is shown in the next subsection.

[2]64-bit counter can solve the overflow problem, however, the project customer allocated only 32-bit for the counter field due to bandwidth limitation and compatibility issues.

D. Message security algorithms

The following algorithms for secure message exchange are implemented on both the DA and the field device.

```
Securing $data$ for sending through channel $C$
 1. Extract Key Table entries for $C$:
      • $SK$: current session key
      • $ctr$: sent-message counter
      • $tag$: key tag value
 2. Construct $D$ = $ctr$ || size($data$) || $data$;
 2. Compute $mac$ = MAC( $SK$, $D$ );
 3. Compute $enc$ = Encrypt( $SK$, IV=0, $D$ || $mac$ );
 4. Send $message$ = $tag$ || size($enc$) || $enc$;
 5. Increment $ctr$;  /* succeeds up to once per ms */
 6. Return OK;
Deciphering $message$ received from $sender$
 1. Parse $message$ = $tag$ || size($enc$) || $enc$;
 2. Deduce the channel $C$ from $tag$ and $sender$;
 3. Extract Key Table entries for $C$:
      • $SK$: current session key
      • $anchorCtr$: anchor counter from latest key update
      • $lastCtr$: last received counter, relative to anchor
      • $recentMAC[10][4]$: recent partial MAC bytes
 4. Compute $dec$ = Decrypt( $SK$, IV=0, $enc$ );
 5. Parse $dec$ = $D$ || $mac$;
 6. Message integrity check:
      a. Compute $mac'$ = MAC( $SK$, $D$ );
      b. If $mac' \neq mac$ return MAC_FAIL;
 7. Parse $D$ = $ctr$ || size($data$) || $data$;
 8. Replay attack detection:
      a. Compute $relCtr$ = $(ctr - anchorCtr) \mod 2^{32}$;
      b. If $relCtr < lastCtr$ return ReplayAttack;
      c. If ($relCtr == lastCtr$ &&
             $mac_{8-11} \in recentMAC$) return ReplayAttack;
      d. Update oldest entry of $recentMAC$ to $mac_{8-11}$;
      e. Update $lastCtr$ = $relCtr$;
 9. If $data$ is a session key update confirmation
        Update $anchorCtr$ = $ctr$;
10. Return OK;
```

V. Conclusions

The proposed solution provides efficient implementation of cryptographic algorithms and key management on resource-constrained devices in smart grid, which will be the building blocks to achieve data confidentiality, integrity, and authentication. Our solution is able to detect and revoke any compromised device, as well as provide efficient, resilient authentication against both insider and outsider attacks.

Acknowledgement

The work was supported by the National Research Foundation (NRF), Prime Minister's Office, Singapore, under its National Cybersecurity R&D Programme (Award No. NRF2014NCR-NCR001-31) and administered by the National Cybersecurity R&D Directorate.

References

[1] International Energy Agency. (2011). Technology Roadmap: Smart Grids, http://www.iea.org/publications/freepublications/publication/smartgrids_roadmap.pdf

[2] D. He, C. Chen, J. Bu, S. Chan, Y. Zhang, and M. Guizani, "Secure Service Provision in Smart Grid Communications," IEEE Commun. Mag., pp. 53-81, Aug. 2012.

[3] M. Mimoso, "Risky Schneider Electric SCADA Vulnerabilities Remain Unpatched," 17 Aug. 2015, https://threatpost.com/risky-schneider-electric-scada-vulnerabilities-remain-unpatched/114305/

[4] Symantec Security Response, "Dragonfly: Western Energy Companies Under Sabotage Threat," 30 Jun. 2014. http://www.symantec.com/connect/blogs/dragonfly-western-energy-companies-under-sabotage-threat

[5] L. Ardito, G. Procaccianti, G. Menga, and M. Morisio, "Smart Grid Technologies in Europe: An Overview", Energies, vol. 6, pp. 251-281, 2013.

[6] X. Li, X. Liang, R. Lu, X. Shen, X. Lin, and H. Zhu, "Securing Smart Grid: Cyber Attacks, Countermeasures, and Challenges," IEEE Commun. Mag., vol. 50, pp. 38-45, 2012.

[7] W. Wang and Z. Lu, "Cyber Security in the Smart Grid: Survey and Challenges," Comput. Netw., 57(5):1344-1371, 2013.

[8] R. Lipovsky, and A. Cherepanov, "BlackEnergy trojan strikes again: Attacks Ukrainian electric power industry," 04 Jan. 2016. http://www.welivesecurity.com/2016/01/04/blackenergy-trojan-strikes-again-attacks-ukrainian-electric-power-industry/

[9] Electric Power Research Institute, "Report to NIST on the Smart Grid Interoperability Standards Roadmap," Palo Alto, CA, Rep. 17 June 2009.

[10] G. Hug and J. A. Giampapa, "Vulnerability Assessment of AC State Estimation With Respect to False Data Injection Cyber-Attacks," IEEE Trans. Smart Grid, 3(3):1362-1370, 2012.

[11] K. C. Sou, H. Sandberg, and K. H. Johansson, "On the Exact Solution to a Smart Grid Cyber-Security Analysis Problem," IEEE Trans. Smart Grid, 4(2):856-865, 2013.

[12] J. Lin, W. Yuy, X. Yang, G. Xuy, and W. Zhao, "On False Data Injection Attacks against Distributed Energy Routing in Smart Grid," IEEE/ACM Int. Conf. Cyber-Physical Systems, pp. 183-192, 2012.

[13] J. Leyden, "Smart Meter SSL Screw-up Exposes Punters' TV Habits," 9 January 2012. http://www.theregister.co.uk/2012/01/09/smart_meter_privacy_oops/

[14] L. Enbysk, "Energy Theft: From Bad to Worse (and What Some Utilities Are Doing About It)," 3 January 2013. http://www.smartgridnews.com/artman/publish/Technologies_Metering/Energy-theft-from-bad-to-worse-and-what-some-utilities-are-doing-about-it-5393.html

[15] Deloitte, "Using Analytics to Crack Down on Electricity Theft," 2 December 2013. http://deloitte.wsj.com/cio/2013/12/02/using-analytics-to-crack-down-on-electricity-theft/

[16] A. A. Cardenas and R. Safavi-Naini, "Security and Privacy in the Smart Grid," in Handbook on Securing Cyber-Physical Critical Infrastructure, Burlington, MA: Morgan Kaufmann, ch. 25, pp.637-654, 2012.

[17] D. Kushner, "The Real Story of Stuxnet," IEEE Spectr., 50(3):48-53, 2013.

[18] M. Badra and S. Zeadally, "Key Management Solutions in the Smart Grid Environment," Joint IFIP Wireless and Mobile Networking Conf. (WMNC), 2013.

[19] International Electrotechnical Commission (IEC), "Smart Grid - Core IEC Standards." http://www.iec.ch/smartgrid/standards/

[20] O. Pal, S. Saiwan, P. Jain, Z. Saquib, and D. Patel, "Cryptographic Key Management for SCADA System: An Architectural Framework," IEEE Int. Conf. Advances in Computing, Control, and Telecommunication Technologies, pp. 169-174, 2009.

[21] F. F. Demertzis, G. Karopoulos, C. Xenakis, and A. Colarieti, "Self-organised Key Management for the Smart Grid," ADHOC-NOW, Lecture Notes in Computer Science 9143, pp. 303-316, 2015.

[22] N. Liu, J. Chen, L. Zhu, J. Zhang, and Y. He, "A Key Management Scheme for Secure Communications of Advanced Metering Infrastructure in Smart Grid," IEEE Trans. Industrial Electronics, 60(1):4746-4756, 2013.

[23] J. Y. Kim and H. K. Choi, "An Efficient and Versatile Key Management Protocol for Secure Smart Grid Communications?" IEEE Wireless Communications and Networking Conference: Mobile and Wireless Networks, pp. 1823-1828, 2012.

[24] R. Prasad and N. Saxena, "Efficient Device Pairing using 'Human-Comparable' Synchronized Audiovisual Patterns," Int. Conf. Applied Cryptography and Network Security (ACNS'08), Lecture Notes In Computer Science 5037, pp. 328-345, 2008.

[25] J. Wiles, "Physical Security: SCADA and the Critical Infrastructure's Biggest Vulnerability," in Techno Security's Guide to Securing SCADA: A Comprehensive Handbook on Protecting the Critical Infrastructure, Burlington, MA: Syngress Publishing, ch. 1, pp. 65, 2008.

[26] K. Mets, J. A. Ojea, and C. Develder, "Combining Power and Communication Network Simulation for Cost-Effective Smart Grid Analysis," IEEE Commun. Surveys & Tutorials, 16(3):1771-1796, 2014.

[27] Pacific Northwest National Laboratory and U.S. Department of Energy, "The Role of Authenticated Communications for Electric Power Distribution," Natl. Workshop - Beyond SCADA: Networked Embedded Control for Cyber Physical Systems, 2006.

[28] L. Lamport, "Password Authentication with Insecure Communication," Commun. ACM, 24(11):770-772, 1981.

[29] H. Alzaid, D.-G. Park, J. G. Nieto, and E. Foo, "Mitigating Sandwich Attacks Against A Secure Key Management in Wireless Sensor Networks for PCS/SCADA," IEEE Int. Conf. Advanced Information Networking and Applications, 2010.

[30] K. Yu, M. Arifuzzaman, Z. Wen, D. Zhang, and T. Sato, "A Key Management Scheme for Secure Communications of Information Centric Advanced Metering Infrastructure in Smart Grid," IEEE Trans. Instrumentation and Measurement, 64(8):2072-2085, 2015.

[31] N. Liu, J. Chen, L. Zhu, J. Zhang, and Y. He, "A Key Management Scheme for Secure Communications of Advanced Metering Infrastructure in Smart Grid," IEEE Trans. Industrial Electronics, 60(10):4746-4756, October 2013.

[32] Z. Wan, G. Wang, Y. Yang, and S. Shi, "SKM: Scalable Key Management for Advanced Metering Infrastructure in Smart Grids," IEEE Trans. Industrial Electronics, 61(12):7055-7066, 2014.

[33] X. Long, D. Tipper, and Y. Qian, "An Advanced Key Management Scheme for Secure Smart Grid Communications," IEEE Int. Conf. Smart Grid Communications, pp. 504-509, 2013.

[34] S. Uludag, K.-S. Lui, W. Ren, and K. Nahrstedt, "Secure and Scalable Data Collection With Time Minimization in the Smart Grid," IEEE Trans. Smart Grid, 7(1):43-54, 2016.

[35] NIST, "Recommendation for Block Cipher Modes of Operation: Methods and Techniques," 2001. http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

[36] IETF, "PKCS #5: Password-Based Cryptography Specification, Version 2.0," 2000. https://tools.ietf.org/html/rfc2898

[37] NIST, "FIPS PUB 113: Standard on Computer Data Authentication," 1985. http://csrc.nist.gov/publications/fips/fips113/fips113.html