Security and Privacy in Smart Grids

Edited by

YANG XIAO
Contents

Preface vii
Acknowledgment ix
About the Editor xi
Contributors xiii

Part 1 Smart Grids in General

Chapter 1 An Overview of Recommendations for a Technical Smart Grid Infrastructure 3
Petra Beenken, Robert Bleiker, José González, Sebastian Rohjans, Michael Specht, Joern Trefke, and Mathias Uslar

Chapter 2 Smart Grid and Cloud Computing: Minimizing Power Consumption and Utility Expenditure in Data Centers 57
Sumit Kumar Bose, Michael Salsburg, Scott Brock, and Ronald Skeoch

Chapter 3 Distributed Opportunistic Scheduling for Building Load Control 85
Peizhong Yi, Xihua Dong, Abiodun Iwayemi, and Chi Zhou
## CONTENTS

### Chapter 4  Advanced Metering Infrastructure and Its Integration with the Distribution Management System  
ZHAO LI, FANG YANG, ZHENYUAN WANG, AND YANZHU YE  

### Chapter 5  Cognitive Radio Network for the Smart Grid  
RAGHURAM RANGANATHAN, ROBERT QIU, ZHEN HU, SHUJIE HOU, ZHE CHEN, MARBIN PAZOS-REVILLA, AND NAN GUO  

### Part 2  Security and Privacy in Smart Grids

#### Chapter 6  Requirements and Challenges of Cybersecurity for Smart Grid Communication Infrastructures  
ROSE QINGYANG HU AND YI QIAN  

#### Chapter 7  Regulations and Standards Relevant for Security of the Smart Grid  
STEFFEN FRIES AND HANS-JOACHIM HOF  

#### Chapter 8  Vulnerability Assessment for Substation Automation Systems  
ADAM HAHN, MANIMARAN GOVINDARASU, AND CHEN-CHING LIU  

#### Chapter 9  Smart Grid, Automation, and SCADA System Security  
YONGGE WANG  

#### Chapter 10  Smart Grid Security in the Last Mile  
TAE OH, SUMITA MISHRA, AND CLARK HOCHGRAF  

### List of Acronyms  

### Index
Preface

A smart grid is an integration of power delivery systems with communication networks and information technology (IT) to provide better services. Security and privacy will provide significant roles in building future smart grids. The purpose of this edited book is to provide state-of-the-art approaches and novel technologies for security and privacy in smart grids covering a range of topics in these areas.

This book investigates fundamental aspects and applications of smart grids, security, and privacy. It presents a collection of recent advances in these areas contributed by many prominent researchers working on smart grids and related fields around the world. Containing 10 chapters divided into two parts—Part I: Smart Grids in General and Part II: Security and Privacy in Smart Grids, we believe this book will provide a good reference for researchers, practitioners, and students who are interested in the research, development, design, and implementation of smart grid security and privacy.

This work is made possible by the great efforts of our contributors and publisher. We are indebted to our contributors, who have sacrificed days and nights to put together these chapters for our readers. We
would like to thank our publisher. Without their encouragement and quality work, we could not have this book.

Yang Xiao
Department of Computer Science
The University of Alabama
Tuscaloosa, Alabama
E-mail: yangxiao@ieee.org
9.3 SCADA Security

In this section, we demonstrate the challenges to secure the current automation systems, such as SCADA systems with examples. Part of these analysis were taken from the work of Wang. In a typical SCADA system, data acquisition and control are performed by remote terminal units (RTUs) and field devices that include functions for communications and signaling. SCADA systems normally use a poll response model for communications with clear text messages. Poll messages are typically small (less than 16 bytes), and responses might range from a short “I am here” to a dump of an entire day’s data. Some SCADA systems may also allow for unsolicited reporting from remote units. The communications between the control center and remote sites could be classified into the following four categories.

1. **Data acquisition**: The control center sends poll (request) messages to RTUs, and the RTUs dump data to the control center. In particular, this includes status scan and measured value scan. The control center regularly sends a status scan request to remote sites to obtain field devices status (e.g., OPEN or CLOSED or a fast CLOSED-OPEN-CLOSED sequence) and a measured value scan request to obtain measured values of field devices. The measured values could be analog values or digitally coded values and are scaled into engineering format by the front-end processor (FEP) at the control center.

2. **Firmware download**: The control center sends firmware downloads to remote sites. In this case, the poll message is larger (e.g., larger than 64,000 bytes) than other cases.

3. **Control functions**: The control center sends control commands to an RTU at remote sites. Control functions are grouped into four subclasses: individual device control (e.g., to turn on/off a remote device); control messages to regulating equipment (e.g., a RAISE/LOWER command to adjust the remote valves); sequential control schemes (a series of correlated individual control commands); and automatic control schemes (e.g., closed control loops).

4. **Broadcast**: The control center may broadcast messages to multiple RTUs. For example, the control center broadcasts an emergent shutdown message or a set-the-clock-time message.
Acquired data are automatically monitored at the control center to ensure that measured and calculated values lie within permissible limits. The measured values are monitored with regard to rate of change and for continuous trend monitoring. They are also recorded for postfault analysis. Status indications are monitored at the control center with regard to changes and time tagged by the RTUs. In legacy SCADA systems, existing communication links between the control center and remote sites operate at very low speeds (could be on an order of 300 to 9,600 bps). Note that present deployments of SCADA systems have variant models and technologies, which may have much better performances (for example, 61850-based systems). Figure 9.1 describes a simple SCADA system.

In practice, more complicated SCADA system configurations exist. Figure 9.2 lists three typical SCADA system configurations (see, e.g., Report No. 12 of the American Gas Association [AGA]21).

Recently, there have been several efforts to secure the national SCADA systems. Examples exist for the following companies and standards:

1. American Gas Association.21 The AGA was among the first to design a cryptographic standard to protect SCADA systems. The AGA had originally been designing a cryptographic standard to protect SCADA communication links; the finished report is AGA 12, part 1. AGA 12, part 2, has been transferred to the Institute of Electrical and Electronics Engineers (IEEE) (IEEE 1711).

2. IEEE 1711.22 This was transferred from AGA 12, part 2. This standard effort tries to define a security protocol, the Serial SCADA Protection Protocol (SSPP), for control system serial communication.
3. IEEE 1815. The standard for Electric Power Systems Communications—Distributed Network Protocol (DNP3). The purpose of this standard is to document and make available the specifications for the DNP3 protocol.


5. National Institute of Standards and Technology (NIST). The NIST Industrial Control System Security (ICS) group works on general security issues related to control systems such as SCADA systems.

6. National SCADA Test Bed Program. The DoE established the National SCADA Test Bed program at Idaho National Laboratory and Sandia National Laboratory to ensure the secure, reliable, and efficient distribution of power.
9.3.1 Threats to SCADA Systems

SCADA systems were not designed with public access in mind; they typically lack even rudimentary security. However, with the advent of technology, particularly the Internet, much of the technical information required to penetrate these systems is widely discussed in the public forums of the affected industries. Critical security flaws for SCADA systems are well known to potential attackers. It is feared that SCADA systems can be taken over by hackers, criminals, or terrorists. Some companies may assume that they use leased lines and therefore nobody has access to their communications. The fact is that it is easy to tap these lines.\(^\text{28}\) Similarly, frequency-hopping spread-spectrum radio and other wireless communication mechanisms frequently used to control RTUs can be compromised as well.

Several efforts\(^{26,27,29}\) have been made for the analysis and protection of SCADA system security. According to these reports,\(^{26,27,29}\) the factors that have contributed to the escalation of risk to SCADA systems include the following:

- The adoption of standardized technologies with known vulnerabilities. In the past, proprietary hardware, software, and network protocols made it difficult to understand how SCADA systems operated—and therefore how to hack into them. Today, standardized technologies such as Windows, Unix-like operating systems, and common Internet protocols are used by SCADA systems. Thus, the number of people with knowledge to wage attacks on SCADA systems has increased.
- The connectivity of control systems to other networks. To provide decision makers with access to real-time information and allow engineers to monitor and control the SCADA systems from different points on the enterprise networks, the SCADA systems are normally integrated into the enterprise networks. Enterprises are often connected to partners’ networks and to the Internet. Some enterprises may also use wide-area networks and the Internet to transmit data to remote locations. This creates further security vulnerabilities in SCADA systems.
• Insecure remote connections. Enterprises often use leased lines, wide-area networks/Internet, and radio/microwave to transmit data between control centers and remote locations. These communication links could be easily hacked.

• The widespread availability of technical information about control systems. Public information about infrastructures and control systems is readily available to potential hackers and intruders. Sean Gorman’s dissertation (see, e.g.,13,18), mentioned previously, is a good example for this scenario. Significant information on SCADA systems is publicly available (from maintenance documents, from former employees, and from support contractors, etc.). All these information sources could assist hackers in understanding the systems and finding ways to attack them.

Hackers may attack SCADA systems with one or more of the following actions:

1. Causing denial-of-service attacks by delaying or blocking the flow of information through control networks
2. Making unauthorized changes to programmed instructions in RTUs at remote sites, resulting in damage to equipment, premature shutdown of processes, or even disabling of control equipment.
3. Sending false information to control system operators to disguise unauthorized changes or to initiate inappropriate actions by system operators
4. Modifying the control system software, producing unpredictable results
5. Interfering with the operation of safety systems

The analysis in reports26,27,29 showed that securing control systems poses significant challenges, which include

1. The limitations of current security technologies in securing control systems. Existing Internet security technologies such as authorization, authentication, and encryption require more bandwidth, processing power, and memory than control system components typically have. Controller stations are generally designed to do specific tasks, and they often use low-cost, resource-constrained microprocessors.
2. The perception that securing control systems may not be economically justifiable.

3. The conflicting priorities within organizations regarding the security of control systems. In this chapter, we concentrate on the protection of SCADA remote communication links. In particular, we discuss the challenges for protection of these links and design new security technologies to secure SCADA systems.

9.3.2 Securing SCADA Remote Connections

Relatively cheap attacks could be mounted on SCADA system communication links between the control center and RTUs since there is neither authentication nor encryption on these links. Under the umbrella of NIST’s Critical Infrastructure Protection Cybersecurity of Industrial Control Systems, the AGA SCADA Encryption Committee has been trying to identify the functions and requirements for authenticating and encrypting SCADA communication links. Their proposal\(^\text{21}\) is to build cryptographic modules that could be invisibly embedded into existing SCADA systems (in particular, one could attach these cryptographic modules to modems, such as those of Figure 9.2) so that all messages between modems are encrypted and authenticated when necessary, and they have identified the basic requirements for these cryptographic modules. However, due to the constraints of SCADA systems, no viable cryptographic protocols have been identified to meet these requirements. In particular, the challenges for building these devices are\(^\text{21}\)

1. Encrypting of repetitive messages.
2. Minimizing delays due to cryptographic operations.
3. Ensuring integrity with minimal latency:
   - Intramessage integrity: If cryptographic modules buffer a message until the message authenticator is verified, it introduces message delays that are not acceptable in most cases.
   - Intermessage integrity: Reorder messages, replay messages, and destroy specific messages.
4. Accommodating various SCADA poll response and retry strategies: Delays introduced by cryptographic modules may
interfere with the SCADA system’s error-handling mechanisms (e.g., time-out errors).
5. Supporting broadcast messages.
6. Incorporating key management.
7. Controlling the cost of devices and management.
8. Dealing with a mixed mode: Some SCADA systems have cryptographic capabilities; others do not.
9. Accommodating different SCADA protocols: SCADA devices are manufactured by different vendors with different proprietary protocols.

Wang\textsuperscript{19} has recently designed efficient cryptographic mechanisms to address these challenges and to build cryptographic modules as recommended in AGA Report No. 12.\textsuperscript{21} These mechanisms can be used to build plug-in devices called sSCADA (secure SCADA) devices that could be inserted into SCADA networks so that all communication links are authenticated and encrypted. In particular, authenticated broadcast protocols are designed so that they can be cheaply included into these devices. It has been a major challenging task to design efficiently authenticated emergency broadcast protocols in SCADA systems.

9.3.3 sSCADA Protocol Suite

The sSCADA protocol suite\textsuperscript{19} is proposed to overcome the challenges discussed in the previous section. A sSCADA device installed at the control center is called a master sSCADA device, and sSCADA devices installed at remote sites are called slave sSCADA devices. Each master sSCADA device may communicate privately with several slave sSCADA devices. Occasionally, the master sSCADA device may also broadcast authenticated messages to several slave sSCADA devices (e.g., an emergency shutdown). An illustrative sSCADA device deployment for point-to-point SCADA configuration is shown in Figure 9.3.

It should be noted that the AGA had originally designed a protocol suite to secure the SCADA systems\textsuperscript{21,30} (an open source implementation could be found in Reference 31). However, Wang\textsuperscript{19} has broken these protocol suites by mounting a replay attack.
To reduce the cost of sSCADA devices and management, only symmetric key cryptographic techniques are used in our design. Indeed, due to the slow operations of public key cryptography, public key cryptographic protocols could introduce delays in message transmission that are not acceptable to SCADA protocols. Semantic security property is used to ensure that an eavesdropper has no information about the plaintext, even if the eavesdropper sees multiple encryptions of the same plaintext. For example, even if the attacker has observed the ciphertexts of “shut down” and “turn on,” it will not help the attacker to distinguish whether a new ciphertext is the encryption of “shut down” or “turn on.” In practice, the randomization technique is used to achieve this goal. For example, the message sender may prepend a random string (e.g., 128 bits for Advanced Encryption Standard [AES] 128) to the message and use special encryption modes such as chaining block cipher (CBC) mode or hash-CBC (HCBC) mode. In some modes, this random string is called the initialization vector (IV). This prevents information leakage from the ciphertext even if the attacker knows several plaintext/ciphertext pairs encrypted with the same key.

Since SCADA communication links could be as low as 300 bps and immediate responses are generally required, there is no sufficient bandwidth to send the random string (IV) each time with the ciphertext; thus, we need to design different cryptographic mechanisms to achieve semantic security without additional transmission overhead. In our design, we use two counters shared between two communicating partners, one for each direction of communication.

The counters are initially set to zeros and should be at least 128 bits, which ensures that the counter values will never repeat, avoiding replay attacks. The counter is used as the IV in message encryptions if CBC or HCBC mode is used. After each message encryption, the counter is increased by one if CBC mode is used, and it is increased by the number of blocks of encrypted data if the HCBC mode is
used. The two communicating partners are assumed to know the values of the counters, and the counters do not need to be added to each ciphertext. Messages may become lost, and the two counters need to be synchronized occasionally (e.g., at off-peak time). A simple counter synchronization protocol is proposed for the sSCADA protocol suite. The counter synchronization protocol could also be initiated when some encryption/decryption errors appear due to unsynchronized counters.

For two sSCADA devices to establish a secure channel, a master secret key needs to be bootstrapped into the two devices at deployment time (or when a new sSCADA device is deployed into the existing network). For most configurations, secure channels are needed only between a master sSCADA device and a slave sSCADA device. For some configurations, secure channels among slave sSCADA devices may also be needed. The secure channel identified with this master secret is used to establish other channels, such as session secure channels, time synchronization channels, authenticated broadcast channels, and authenticated emergency channels.

Assume that $\mathcal{H}(\cdot)$ is a pseudorandom function (e.g., constructed from Secure Hash Algorithm [SHA]-256) and two sSCADA devices $A$ and $B$ share a secret $\mathcal{K}_{AB} = \mathcal{K}_{BA}$. Depending on the security policy, this key $\mathcal{K}_{AB}$ could be the shared master secret or a shared secret for one session that could be established from the shared master key using a simple key establishment protocol (to achieve session key freshness, typically one node sends a random nonce to the other one, and the other node sends the encrypted session key together with an authenticator on the ciphertext and the random nonce). Keys for different purposes could be derived from this secret as follows (it is not a good practice to use the same key for different purposes): For example, $K_{AB} = \mathcal{H}(\mathcal{K}_{AB}, 1)$ is for message encryption from $A$ to $B$, $K'_{AB} = \mathcal{H}(\mathcal{K}_{AB}, 2)$ is for message authentication from $A$ to $B$, $K_{BA} = \mathcal{H}(\mathcal{K}_{AB}, 3)$ is for message encryption from $B$ to $A$, and $K'_{BA} = \mathcal{H}(\mathcal{K}_{AB}, 4)$ is for message authentication from $B$ to $A$.

Optional message authentication codes (MACs) are used for two parties to achieve data authentication and integrity. MACs that could be used for sSCADA implementation include HMAC, CBC-MAC, and others. When party $A$ wants to send a message $m$ to party $B$ securely, $A$ computes the ciphertext $c = \mathcal{E}(C_A, K_{AB}, \overline{c_A} \ || \ m)$ and message authenticator $mac = MAC(K'_{AB}, C_A \ || \ c)$, where $\overline{c_A}$ is
the last $l$ bits of $\mathcal{H}(C_A)$ ($l$ could be as large as possible if bandwidth is allowed, and 32 bits should be the minimal), $E(C_A, K_{AB}, \overline{C}_A || m)$ denotes the encryption of $\overline{C}_A || m$ using key $K_{AB}$ and random-prefix (or IV) $C_A$, and $C_A$ is the counter value for the communication from $A$ to $B$. Then, $A$ sends the following packets to $B$:

$$A \rightarrow B: \ c, mac \ (\text{optional})$$

When $B$ receives these packets, $B$ decrypts $c$, checks that $\overline{C}_A$ is correct, and verifies the message authenticator $mac$ if $mac$ is present. As soon as $B$ receives the first block of the ciphertext, $B$ can check whether $\overline{C}_A$ is correct. If it is correct, then $B$ continues the decryption and updates its counter. Otherwise, $B$ discards the entire ciphertext. If the message authenticator code $mac$ is present, $B$ also verifies the correctness of $mac$. If $mac$ is correct, $B$ does nothing; otherwise, $B$ may choose to inform $A$ that the message was corrupted or try to resynchronize the counters.

There are several implementation issues on how to deliver the message to the target (e.g., RTU). For example, there are the following:

1. $B$ uses the counter to decrypt the first block of the ciphertext; if the first $l$ bits of the decrypted plaintext are not consistent with $\mathcal{H}(C_A)$, then the reason could be that the counter $C_A$ is not synchronized or that the ciphertext is corrupted. $B$ may try several possible counters until the counter-checking process succeeds. $B$ then uses the verified counter and the corresponding key to decrypt the message and deliver each block of the resulting message to the target as soon as it is available. If no counter could be verified in a limited number of trials, $B$ may notify $A$ of the transmission failure and initiate the counter synchronization protocol in the next section. The advantage of this implementation is that we have minimized delay from the cryptographic devices, thus minimizing the interference of SCADA protocols. Note that in this implementation, the message authenticator $mac$ is not used. If the ciphertext was tampered, we rely on the error correction mechanisms (normally CRC codes) in SCADA systems to discard the entire message. If CBC (respectively
HCBC) mode is used, then the provable security properties (respectively provable online cipher security properties) of CBC mode (respectively HCBC mode)\cite{36,37} guarantee that the attacker has no chance to tamper with the ciphertext, so that the decrypted plaintext contains a correct CRC that was used by SCADA protocols to achieve integrity.

2. Proceed as in case 1. In addition, the \textit{mac} is further checked, and the decrypted message is delivered to the SCADA system only if the \textit{mac} verification passes. The disadvantage for this implementation is that these cryptographic operations introduce significant delay for message delivery, and it may interfere with SCADA protocols.

3. Proceed as in case 1. The decrypted message is delivered to the SCADA system as soon as available. After receiving the entire message and \textit{mac}, \textit{B} will also verify \textit{mac}. If the verification passes, \textit{B} will do nothing. Otherwise, \textit{B} resynchronizes the counter with \textit{A} or initiates some other exception-handling protocols.

4. To avoid delays introduced by cryptographic operations and to check the \textit{mac} at the same time, sSCADA devices may deliver decrypted bytes immediately to the target except the last byte. If the message authenticator \textit{mac} is verified successfully, the sSCADA device delivers the last byte to the target; otherwise, the sSCADA device discards the last byte or sends a random byte to the target. That is, we rely on the error correction mechanisms at the target to discard the entire message. Similar mechanisms have been proposed.\cite{21} However, an attacker may insert garbage between the ciphertext and \textit{mac}, thus tricking the sSCADA device to deliver the decrypted messages to the SCADA system. If this happens, we essentially do not receive an advantage from this implementation. Thus, this implementation is not recommended.

5. Instead of prepending \(c_A\) to the plaintext message, one may choose to prepend three bytes of other specially formatted string to the plaintext message (bandwidth of three bytes is normally available in SCADA systems) before encryption. This is an acceptable solution although we still prefer our solution of prepending the hash outputs of the counter.
There could be other implementations to improve the performance and interoperability with SCADA protocols. sSCADA devices should provide several possible implementations for users to configure. Indeed, sSCADA devices may also be configured in a dynamic way so that for different messages it uses different implementations.

In some SCADA communications, message authentication only is sufficient. That is, it is sufficient for \( A \) to send \((m, mac)\) to \( B \), where \( m \) is the cleartext message and \( mac = MAC(K'_{AB}, C_A \| m) \). sSCADA devices should provide configuration options to perform message authentication without encryption. In this case, even if the counter value is not used as the IV, the counter value should still be authenticated in the \( mac \) and be increased after the operation. This will provide message freshness assurance and avoid replay attacks. sSCADA should also support message pass-through mode. That is, the message is delivered without encryption and authentication. In summary, it should be possible to configure an sSCADA device in such a way that some messages are authenticated and encrypted, some messages are authenticated only, and some messages are passed through directly.

### 9.3.4 Counter Synchronization

In the point-to-point message authentication and encryption protocol, we assume that both sSCADA devices \( A \) and \( B \) know each other’s counter values \( C_A \) and \( C_B \), respectively. In most cases, reliable communication in SCADA systems is provided, and the security protocols in the previous section work fine. Still, we provide a counter synchronization protocol so that sSCADA devices can synchronize their counters when necessary. The counter synchronization protocol could be initiated by either side. Assume that \( A \) initiates the counter synchronization protocol. Then, the protocol looks as follows:

\[
A \rightarrow B : \quad N_A \\
B \rightarrow A : \quad C_B, MAC(K'_{BA}, N_A \| C_B)
\]

The initial counter values of two sSCADA devices could be bootstrapped directly. The counter synchronization protocol presented could also be used by two devices to bootstrap the initial counter values. A master sSCADA device may also use the authenticated broadcast
channel that we discuss in the next section to set the counters of several slave sSCADA devices to the same value using one message.

9.4 Conclusion

In this chapter, we discussed the challenges for smart grid system security. We then use control systems (in particular, SCADA systems) as examples for studying how to address these challenges. In particular, we mentioned Wang’s attack\(^\text{19}\) on the protocols in the first version of the AGA standard draft.\(^\text{30}\) This attack showed that the security mechanisms in the first draft of the AGA standard protocol could be easily defeated. We then proposed a suite of security protocols optimized for SCADA/DCS systems. These protocols are designed to address the specific challenges of SCADA systems.

Recently, there has been a wide interest in the secure design and implementation of smart grid systems.\(^\text{38}\) The SCADA system is one of the most important legacy systems of the smart grid systems. Together with other efforts such as those offered in IEEE 1711,\(^\text{22}\) IEEE 1815,\(^\text{23}\) IEC TC 57,\(^\text{24}\) IEC 60870-5,\(^\text{25}\) NIST Industrial Control System Security,\(^\text{26}\) and the National SCADA Testbed Program,\(^\text{27}\) the work in this chapter presents an initial step for securing the SCADA section of the smart grid systems against cyber attacks.

References


