Chapter 1

Smart Grid, Automation, and SCADA Systems Security

Yongge Wang

Department of Software and Information Systems
UNC Charlotte, 9201 University City Blvd., NC 28223, USA
yongge.wang@uncc.edu

In this Chapter, we discuss the challenges for secure smart energy grid and automation systems. We first describe the current security status and existing attacks on power grid and critical infrastructures. Then we use the SCADA system as an example to show the challenges to secure the automation systems and smart power grid systems. Distributed control systems (DCS) and supervisory control and data acquisition (SCADA) systems were developed to reduce labor costs, and to allow system-wide monitoring and remote control from a central location. Control systems are widely used in critical infrastructures such as smart electric grid, natural gas, water and wastewater industries. While control systems can be vulnerable to a variety of types of cyber attacks that could have devastating consequences, little research has been done to secure the control systems. American Gas Association (AGA), IEC TC57 WG15, IEEE, NIST and National SCADA Test Bed Program have been actively designing cryptographic standard to protect SCADA systems. In this chapter, we briefly review these efforts and discuss related security issues.

1. Energy grid and SCADA: a high level introduction

As stated in DOE smart grid white paper,[1] United States is in the process of the Nation's electricity transmission and distribution system modernization "to maintain a reliable and secure electricity infrastructure that can meet future demand growth". The major characterizations[1] of a modern electrical grid system include:

- Improved reliability, security, and efficiency of energy distribution based on modern digital communication and control techniques.

- Integration of industries involved in production and sale of energy, including the gas industry (e.g., natural gas extraction and distribution systems), the electrical power industry, the coal industry, and the renewable resources (e.g., solar and wind power).
- Integration of demand-response technologies such as real-time, automated, interactive technologies that optimize the physical operation of appliances and consumer devices for energy generation, transmission, distribution, and retailing (e.g., metering).
- Deployment of advanced electricity storage and peak-shaving technologies.
- Availability of real time information and control options to consumers.
- Integration of cyber-security techniques within the grid systems.

In a summary, the smart grid system is a secure and intelligent energy distribution system that delivers energy from suppliers to consumers based on two-way demand and response digital communication technologies to control appliances at consumers' homes to save energy and increase reliability. The smart grid system overlays the existing energy distribution system with digital information management and advanced metering systems. It is obvious that the increased interconnection and automation over the grid systems presents new challenges for deployment and management.

It is challenging to securely and efficiently convert the existing power grid systems to a smart system with the above characteristics. According to US Energy Information Administration website,[2] at the end of 2010, there are more than 9200 electric generating plants in USA, including coal, petroleum liquids, petroleum coke, natural gas, other gases, nuclear, hydroelectric, renewables, hydroelectric pumped storage, and others. These generating plants produce 312,334,000 megawatt-hours electricity during February 2011. The electricity are distributed to the consumers via more than 300,000 miles of transmission lines throughout the USA. These power infrastructure was designed for performance rather than security and the integrated communications protocols were designed for bandwidth efficiency without the consideration of cyber security. When moving the current energy distribution infrastructure towards a smart grid, we have to overcome the challenges of integrating network based security solutions with automation systems which usually requires a combination of new and legacy components and may not have enough reserved resource to perform security functionalities. In this chapter, we will use SCADA as an example to il-

lustrate the strategies that may be employed for the design of smart grid systems.

Control systems are computer-based systems that are used within many critical infrastructures and industries (e.g., electric grid, natural gas, water, and wastewater industries) to monitor and control sensitive processes and physical functions. In order to deploy the smart grid system, there is a trend towards interconnecting SCADA systems and data networks (e.g., Intranet). Thus without a secure SCADA system it is impossible to deploy the intelligent smart grid systems.

Typically, control systems collect sensor measurements and operational data from the field, process and display this information, and relay control commands to local or remote equipments. Control systems may perform additional control functions such as operating railway switches, circuit breakers, and adjusting valves to regulate flow in pipelines. The most sophisticated ones control devices and systems at an even higher level.

Control systems have been in place since the 1930s and there are two primary types of control systems. Distributed Control Systems (DCS) and Supervisory Control and Data Acquisition (SCADA) systems. DCS systems typically are used within a single processing or generating plant or over a small geographic area. SCADA systems typically are used for large, geographically dispersed distribution operations. For example, a utility company may use a DCS to generate power and a SCADA system to distribute it. We will concentrate on SCADA systems and our discussions are generally applicable to DCS systems.

2. Recent attacks and accidents with energy systems and automation systems

Several (real and simulated) attacks on energy and SCADA systems were reported in the past few years.[3–13] In the Maroochy Shire attack[3] of the year 2000, an Australian man hacked into the Maroochy Shire, Queensland computerized waste management system and caused 200,000 gallons of raw sewage to spill out into local parks, rivers and even the grounds of a Hyatt Regency hotel. It is reported that the 49-year-old Vitek Boden had conducted a series of electronic attacks on the Maroochy Shire sewage control system after his job application had been rejected. Later investigations found radio transmitters and computer equipments in Boden's car. The laptop hard drive contained software for accessing and controlling the sewage SCADA systems.

By exploiting a vulnerability in a control system, the simulated Aurora Generator Test[5] conducted in March 2007 by the U.S. Department of Homeland Security resulted in hacker's remote access to the generator room at the Idaho National Laboratory and the partial destruction of a $1 million diesel-electric generator destroyed.

In September 2007, an individual who claims himself a CUPE (Canadian Union of Public Employees) member hacked into Vancouver city's computer system that commands the town's traffic lights and set the computer clock seven-hours behind.[6] The result was that traffic signals geared for midnight time were managing traffic for the morning rush hour.

On April 8, 2009, an article[7] in the Wall Street Journal by Gorman reported that "cyberspies have penetrated the U.S. electrical grid and left behind software programs that could be used to disrupt the system, according to current and former national-security officials". The same article mentioned that, instead of damaging the power grid or other key infrastructure, the goals of these attacks were to navigate the U.S. electrical system and its controls for mapping purpose. To make things worse, these attacks were mainly detected by U.S. intelligence agencies instead of the companies in charge of the infrastructure. In another word, the US utility companies are not ready for the protection of their current infrastructure, let alone the future interconnected smart grid systems. These attacks increase worries about cyber attackers that may take control of electrical facilities, a nuclear power plant, financial networks, water, sewage and other infrastructure systems via the Internet.

On Thursday, August 14, 2003, at approximately 4:11 p.m., a widespread power outage occurred throughout parts of the Northeastern and Midwestern United States and Ontario, Canada. According to the New York Independent System Operator (NYISO)'s report,[8] this Northeast Blackout of 2003 affected approximately 10 million people in Ontario and 45 million people in eight U.S. states, and the NYISO MW load had a loss of 80% at the height of the outage. The final report[14] by US-Canada Power System Outage Task Force shows that the blackout was triggered by a race-condition software bug in General Electric Energy's Unix-based XA/21 energy management system. The bug caused a disruption of service at FirstEnergy's control room and the alarm system there stopped working for over an hour. After the alert system failure, neither audio nor visual alerts for important changes in system state are available to the operators. The unprocessed events queued up quickly and the primary server failed within 30 minutes. Then the server applications (including the failed alert

systems) were automatically transferred to the backup server, which failed soon after. The lack of alarms led operators to dismiss a call from American Electric Power (AEP) about the tripping and reclosure of a 345 kV shared line in northeast Ohio. FirstEnergy's Technical support informed control room operators of the alarm system just before the massive blackout started.[15] Though the software bug triggered this blackout, the US-Canada Power System Outage Task Force report[14] listed four major causes for the blackout:

(1) FirstEnergy and its reliability council "failed to assess and understand the inadequacies of FEs system, particularly with respect to voltage instability and the vulnerability of the Cleveland-Akron area, and FE did not operate its system with appropriate voltage criteria".
(2) FirstEnergy "did not recognize or understand the deteriorating condition of its system".
(3) FirstEnergy "failed to manage adequately tree growth in its transmission rights-of-way".
(4) the "failure of the interconnected grids reliability organizations to provide effective real-time diagnostic support."

The affected infrastructure of the Blackout includes: Power generation (power plants automatically went into "safe mode" to prevent damage in the case of an overload), water supply (some areas lost water pressure because pumps didn't have power), transportation (trains had no power and passenger security checking at affected airports ceased), communication systems (cellular communication devices were disrupted, radio stations were momentarily knocked off the air, and cable television systems were disabled), manufacturing (large numbers of factories were closed in the affected area and freeway congestion in affected areas affected the "just-in-time" supply system).

In June 2010, it was reported[9,16] that Stuxnet worm sp reads around the world (with 59% infected systems in Iran) to subvert SCADA systems. Stuxnet malware targets only Siemens SCADA applications PCS 7, WinCC and STEP7 that run on Microsoft Windows and Siemens S7 programmable logic controller (PLC). The worm initially spreads using USB flash drives and then uses four zero-day exploits to infect Siemens SCADA and HMI (Human Machine Interface) system SIMATIC WinCC and PCS 7. Once infected, it attacks PLC systems with variable-frequency drives that spin between 807Hz and 1210Hz. When certain criteria are met, Stuxnet periodically modifies the frequency to 1410Hz and then to 2Hz and then to

1064Hz, and thus affects the operation of the connected motors by changing their rotational speed.

In the 2009 Black Hat conference at Las Vegas, Mike Davis[10] showed a simulation environment in which an attacker could take control of 15,000 out of 22,000 home smart meters within 24 hours by exploiting design flaws within an unnamed brand of smart meters.

Since November 2009, there have been reported[11] coordinated covert and targeted cyberattacks against global oil, energy, and petrochemical companies. These attacks are named as the Night Dragon attack by McAfree.[11] The attack first compromises company extranet web servers through SQL-injection techniques and then uploads some commonly available hacker tools to the compromised web servers, which will allow the attacker to break into the company's intranet and get access to some sensitive internal desktops and servers. By disabling Microsoft Internet Explorer (IE) proxy settings, the attacker achieves direct communication from infected machines to the Internet. The attacker proceeds further to connect to other machines (targeting executives) and exfiltrating email archives and other sensitive documents.

According to Zetter,[12] in May 2011, NSS Lab[17] researchers only spent two months of times on a few SCADA devices and found several vulnerabilities in Siemens PLC and SCADA control systems that could be exploitd by hackers to get remote access to the control systems to cause physical destruction to factories and power plants. It should be noted that Siemens PLC and SCADA systems are widely used in the world controlling critical infrastructure systems such as nuclear power and enrichment plants and commercial manufacturing facilities. Under the pressure by the Department of Homeland Security, the NSS lab did not disclose details before Siemens could patch the vulnerabilities. This example shows that when the control systems are interconnected with the intranet, a dedicated attacker could easily mount serious attacks. It should also noted that, in his dissertation, the PhD student Sean Gorman from George Mason University mapped every business and industrial sector in the American economy to the fiber-optic network that connects them, using materials that was available publicly on the Internet (see, e.g.,[13,18]). Similarly, under the pressure from the government, Gorman's dissertation has never been made public.

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 are taken from Wang.[19] In a typical SCADA system,[20] data acquisition and control are performed by remote terminal units (RTU) 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 following four categories.

(1) *Data acquisition*: the control center sends poll (request) messages to remote terminal units (RTU) 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 get field devices status (e.g., OPEN or CLOSED or a fast CLOSED-OPEN-CLOSED sequence) and a measured value scan request to get 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 64K bytes) than other cases.
(3) *Control functions*: the control center sends control commands to a 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 remote terminal units (RTUs). For example, the control center broadcasts an emergent shutdown message or a set-the-clock-time message.

Acquired data is 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 post-fault 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 300bps to 9600bps). Note that present deployments of SCADA systems have variant models and technologies, which may have much better performances (for example, 61850-based systems). Figure 1 describes a simple SCADA system.

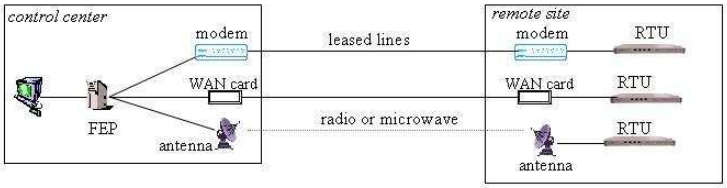

Fig. 1. A simple SCADA system

In practice, more complicated SCADA system configurations exist. Figure 2 lists three typical SCADA system configurations (see, e.g., AGA Report No. 12[21]).

Recently, there have been several efforts to secure the national SCADA systems. The examples are:

(1) American Gas Association (AGA).[21] AGA is among the first to design cryptographic standard to protect SCADA systems. American Gas Association (AGA) had originally been designing cryptographic standard to protect SCADA communication links and finished the report AGA 12 part 1. The AGA 12 part 2 has been transferred to IEEE 1711.
(2) IEEE 1711.[22] This is 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.[23] Standard for Electric Power Systems Communications – Distributed Network Protocol (DNP3). The purpose of this standard

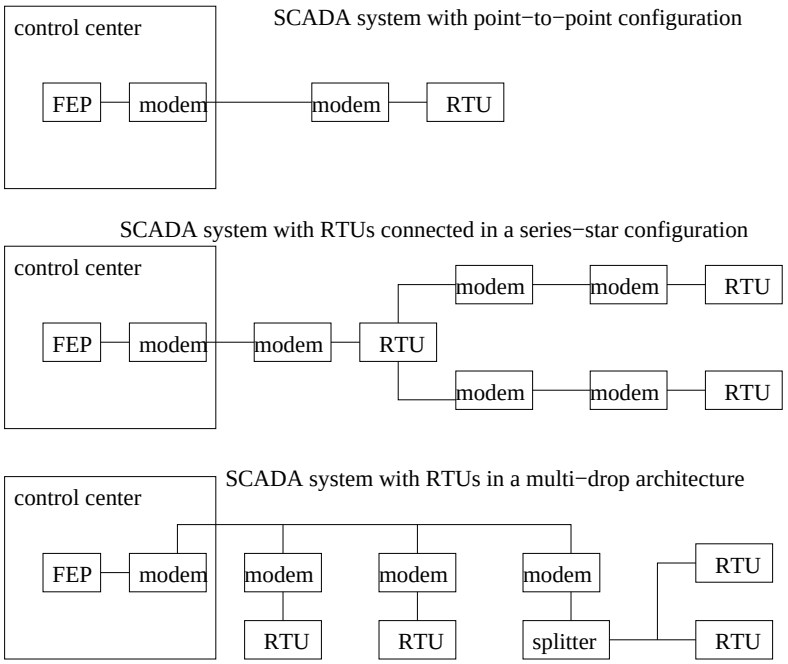

Fig. 2. Typical SCADA system configurations

is to document and make available the specifications for the DNP3 protocol.

(4) IEC TC57 WG15.[24,25] IEC TC57 WG57 standardize SCADA communication security via its IEC 608705 series.
(5) NIST.[26] 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.[27] The Department of Energy established the National Supervisory Control and Data Acquisition (SCADA) Test Bed program at Idaho National Laboratory and Sandia National Laboratory to ensure the secure, reliable and efficient distribution of power.

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 tech-

nology and 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.[28] Similarly, frequency hopping spread spectrum radio and other wireless communication mechanisms frequently used to control remote terminal units (RTU) can be compromised as well.

Several efforts[26,27,29] have been put on 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 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 have increased.
- The connectivity of control systems to other networks. In order to provide decision makers with access to real-time information and allowing 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 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]) that we have mentioned earlier is a very 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 could assist hackers in understanding the systems and to find ways to attack them.

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

(1) Denial of service attacks by delaying or blocking the flow of information through control networks.
(2) Make unauthorized changes to programmed instructions in RTUs at remote sites, resulting in damage to equipment, premature shutdown of processes, or even disabling control equipment.
(3) Send false information to control system operators to disguise unauthorized changes or to initiate inappropriate actions by system operators.
(4) Modify the control system software, producing unpredictable results.
(5) Interfere with the operation of safety systems.

The analysis in reports such as[26,27,29] show 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; and
(3) the conflicting priorities within organizations regarding the security of control systems. In this paper, we will concentrate on the protection of SCADA remote communication links. In particular, we discuss the challenges on protection of these links and design new security technologies to secure SCADA systems.

3.2. *Securing SCADA remote connections*

Relatively cheap attacks could be mounted on SCADA system communication links between the control center and remote terminal units (RTU) since there is neither authentication nor encryption on these links. Under the umbrella of NIST "Critical Infrastructure Protection Cybersecurity of Industrial Control Systems", "American Gas Association (AGA)

SCADA Encryption Committee" has been trying to identify the functions and requirements for authenticating and encrypting SCADA communication links. Their proposal[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 of Figure 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:[21]

(1) encryption of repetitive messages
(2) minimizing delays due to cryptographic operations
(3) assuring integrity with minimal latency
    - intra-message integrity: if cryptographic modules buffer message until the message authenticator is verified, it introduces message delays that are not acceptable in most cases
    - inter-message 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) cost of device and management
(8) mixed mode: some SCADA systems have cryptographic capabilities while others not
(9) accommodate to different SCADA protocols: SCADA devices are manufactured by different vendors with different proprietary protocols.

Wang[19] has recently designed efficient cryptographic mechanisms to address these challenges and to build cryptographic modules as recommended in AGA Report No. 12.[21] These mechanisms can be used to build plug-in devices called sSCADA (secure SCADA) 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.

3.3. *sSCADA protocol suite*

The sSCADA protocol suite[19] is proposed to overcome the challenges that we have discussed in the previous section. sSCADA devices that are installed at the control center is called master sSCADA device, and sSCADA devices that are installed at remote sites are called slave sSCADA devices. Each master sSCADA device may communicate privately with several slave sSCADA devices. Once in a while, 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 3.

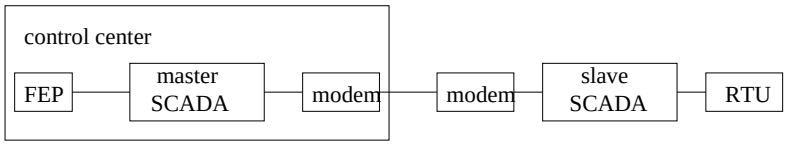

Fig. 3. sSCADA with point-to-point SCADA configuration

It should be noted that American Gas Association had originally designed a protocol suite to secure the SCADA systems[21,30] (an open source implementation could be found at[31]). However, Wang[19] has broken these protocol suites by mounting a replay attack.

In order to reduce the cost of sSCADA devices and management, only symmetric key cryptographic techniques is used in our design. Indeed, due to the slow operations of public key cryptography, public key cryptographic protocols could introduce delays in message transmission which are not acceptable to SCADA protocols. Semantic security property[32] is used to ensure that an eavesdropper has no information about the plaintext, even if it 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 AES-128) to the message and use special encryption modes such as chaining block cipher mode (CBC) or Hash-CBC mode (HCBC). In some mode, 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 300bps and immediate response 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 initialization vector (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 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 get lost and the two counters need to be synchronized once a while (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.

In order for two sSCADA devices to establish a secure channel, a master secret key needs to be bootstrapped into the two devices at the 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 be needed also. 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 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 which could be established from the shared master key using a simple key establishment protocol (in order to achieve session key freshness, typically one node sends a random nonce to the other one and the other node sends the encrypted ses-

sion 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 (MAC) are used for two parties to achieve data authentication and integrity. Message authentication codes that could be used for sSCADA implementation include HMAC,[33,34] CBC-MAC,[35] and others. When party A wants to send a message m to party B securely, A computes the ciphertext $c = \mathcal{E}(C_A, K_{AB}, \bar{c_A}||m)$ and message authenticator $mac = MAC(K'_{AB}, C_A||c)$, where $\bar{c_A}$ 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), $\mathcal{E}(C_A, K_{AB}, \bar{c_A}||m)$ denotes the encryption of $\bar{c_A}||m$ 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 the above packets, B decrypts c, checks that $\bar{c_A}$ 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 $\bar{c_A}$ is correct. If it is correct, then B continues the decryption and updates it 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 re-synchronize the counters.

There are several implementation issues on how to deliver the message to the target (e.g., RTU). For example, we give a few cases in the following.

(1) B uses the counter to decrypt the first block of the ciphertext, if the first l bits of the decrypted plaintext is 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 minimize the interference of SCADA protocols. Note that in this implementation, the message authenticator *mac* is not used at all. 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 on-line cipher security properties) of CBC mode (respectively HCBC mode)[36,37] guarantees that the attacker has no chance to tamper the ciphertext so that the decrypted plaintext contains correct CRC that was used by SCADA protocols to achieve integrity.

(2) Proceed as in the above case 1. In addition, the *mac* is further checked and the decrypted message is delivered to the SCADA system only if the *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 the above case 1. The decrypted message is delivered to the SCADA system as soon as they are available. After receiving the entire message and *mac*, B will also verify *mac*. If the verification passes, B do nothing. Otherwise, B re-synchronizes the counter with A or initiates some other exception handling protocols.

(4) In order to avoid delays introduced by cryptographic operations and to check the *mac* at the same time, sSCADA devices may deliver decrypted bytes immediately to the target except the last byte. If the message authenticator *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 in.[21] However, an attacker may insert garbages between the ciphertext and *mac* thus trick the sSCADA device to deliver the decrypted messages to the SCADA system. If this happens, we essentially do not get advantage from this implementation. Thus this implementation is not recommended.

(5) Instead of prepend $\bar{c_A}$ to the plaintext message, one may choose to prepend three bytes of other specially formated string to the plaintext message (three bytes bandwidth is normally available in SCADA systems) before encryption. This is an acceptable solution though 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 device should provide several possible implementations for users to configure. Indeed, sSCADA devices may also be configured in a dynamic way 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 device should provide configuration options to do 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, message is delivered without encryption and authentication. In a 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.

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. 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 could 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:

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

The initial counter values of two sSCADA devices could be bootstrapped directly. The above counter synchronization protocol 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 will discuss in the next section to set several slave sSCADA devices' counters to the same value using one message.

4. Conclusion

In this chapter, we discussed the challenges for smart grid system security. We then use control systems (in particular, SCADA systems) as example to study how to address these challenges. In particular, we mentioned Wang's attack[19] on the protocols in the first version of AGA standard draft.[30] This attack shows 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 that SCADA systems have.

Recently, there has been a wide interest for the secure design and implementation of smart grid systems.[38] SCADA system is one of the most important legacy systems of the smart grid systems. Together with other efforts such as,[22–27] the works in this chapter present an initial step for securing the SCADA section of the smart grid systems against cyber attacks.

References

1. Department of Energy. Title XIII – Smart Grid, (2010). http://www.oe.energy.gov/DocumentsandMedia/EISA_Title_XIII_Smart_Grid.pdf.
2. US Energy Information Adminitration. Net generation by energy source: Total (all sectors), (2011). http://www.eia.gov/cneaf/electricity/epm/table1_1.html.
3. M. Abrams and J. Weiss. Malicious control system cyber security attack case study–maroochy water services, australia, (2010). http://csrc.nist.gov/groups/SMA/fisma/ics/documents/Maroochy-Water-Services-Case-Study_briefing.pdf.
4. M. Abrams and J. Weiss. Bellingham, washington, control system cyber security case study, (2007). http://csrc.nist.gov/groups/SMA/fisma/ics/documents/Bellingham_Case_Study_report%2020Sep071.pdf.
5. USA Today. AURORA case: US video shows hacker hit on power grid., (2007). http://www.usatoday.com/tech/news/computersecurity/2007-09-27-hacker-video_N.htm.
6. SPAMfighter. Vancouver city - police investigating possible sabotage of traffic light computer system, (2007). http://www.spamfighter.com/News_Show_Other.asp?M=10&Y=2007.
7. S. Gorman. Electricity grid in us penetrated by spies, (2009). http://online.wsj.com/article/SB123914805204099085.html.
8. ISO New York Independent System Operator. Nyiso interim report on the august 14, 2003 blackout, (2004). http://www.hks.harvard.edu/hepg/Papers/NYISO.blackout.report.8.Jan.04.pdf.
9. G. Keizer. Is stuxnet the 'best' malware ever?, (2010). http://www.infoworld.com/print/137598.

10. M. Davis. Smartgrid device security adventures in a new medium, (2009). http://www.blackhat.com/presentations/bh-usa-09/MDAVIS/BHUSA09-Davis-AMI-SLIDES.pdf.
11. McAfee. Global energy cyberattacks: Night dragon, (February 2011). http://www.mcafee.com/us/resources/white-papers/wp-global-energy-cyberattacks-night-dragon.pdf.
12. K. Zetter. Fearing industrial destruction, researcher delays disclosure of new siemens scada holes, (2011). http://www.wired.com/threatlevel/2011/05/siemens-scada-vulnerabilities/.
13. L. Blumenfeld. Dissertation could be security threat. washington post. http://www.washingtonpost.com/ac2/wp-dyn/A23689-2003Jul7.
14. US-Canada Power System Outage Task Force. Final report on the August 14, 2003 Blackout in the United States and Canada: Causes and REcommendations, (April, 2004). https://reports.energy.gov/BlackoutFinal-Web.pdf.
15. North American Electric Reliability Council. Technical analysis of the august 14, 2003, blackout: What happened, why, and what did we learn?, (2004). http://www.nerc.com/docs/docs/blackout/NERC_Final_Blackout_Report_07_13_04.pdf.
16. N. Falliere, L. Murchu, and E. Chien. W32.stuxnet dossier, (February, 2011). http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf.
17. NSSLAB. http://www.nsslabs.com/.
18. J. Rappaport. What you don't know might hurt you: Alum's work balances national security and information sharing. http://gazette.gmu.edu/articles/11144.
19. Y. Wang, sSCADA: Securing SCADA infrastructure communications, *Int. J. Communication Networks and Distributed Systems*. **6**(1), 59–78, (2011).
20. T. Cegrell, *Power System Control Technology*. (Prentice-Hall International (UK) Ltd., 1986).
21. AGA Report No. 12. Cryptographic protection of scada communications: General recommendations. draft 2, february 5, 2004. the draft 2 is no longer available online. the draft 3 is available for purchase at, (2010). http://www.aga.org/.
22. IEEE 1711. Trial use standard for a cryptographic protocol for cyber security of substation serial links, (2011). http://standards.ieee.org/findstds/standard/1711-2010.html.
23. IEEE 1815. Standard for Electric Power Systems Communications - Distributed Network Protocol (DNP3), (2010). http://grouper.ieee.org/groups/1815/.
24. IEC TC 57. Focus on the iec tc 57 standards, (2010). http://www.ieee.org/portal/cms_docs_pes/pes/subpages/publications-folder/TC_57_Column.pdf.
25. IEC60870-5. Group maillist information, (2010). http://www.trianglemicroworks.com/iec60870-5/index.htm.
26. NIST. Nist industrial control system security (ics), (2011). http://csrc.

nist.gov/groups/SMA/fisma/ics/index.html.
27. Idaho National Laboratory. National scada testbed program, (2011). http://www.inl.gov/scada/.
28. Granite Island Group. Wiretapping and outside plant security - wiretapping 101, (2011). http://www.tscm.com/outsideplant.html.
29. GAO-04-628. Critical infrastructure protection: challenges and efforts to secure control systems. testimony before the subcommittee on technology information policy, intergovernmental relations and the census, house committee on government reform, (March 30, 2004). http://www.gao.gov/new.items/d04628t.pdf.
30. A. K. Wright, J. A. Kinast, and J. McCarty., *Low-Latency Cryptographic Protection for SCADA Communications*, In *Proc. 2nd Int. Conf. on Applied Cryptography and Network Security, ACNS 2004*, vol. 3809, *LNCS*, pp. 263–277. Springer Verlag, (2004).
31. A. Wright. Scadasafe, (2006). http://scadasafe.sourceforge.net.
32. S. Goldwasser and S. Michali, Probabilistic encryption, *Journal of Computer and System Sciences*. **28**, 270–299, (1984).
33. M. Bellare, R. Canetti, and H. Krawczyk., Message authentication using hash functions–the hmac construction, *RSA Laboratories CryptoBytes*. **2**, (1996).
34. H. Krawczyk, M. Bellare, and R. Canetti. HMAC: keyed-hashing for message authentication, internet rfc 2104, (February 1997).
35. NIST. Des model of operation, fips publication 81 (fips pub 81), (1981).
36. M. Bellare, A. Boldyreva, L. Knudsen, and C. Namprempre. On-line ciphers and the hash-cbc constructions. In *Advances in Cryptology - Crypto 2001*, vol. 2139, *LNCS*. Springer Verlag, (2001).
37. M. Bellare, J. Kilian, and P. Rogaway, The security of the cipher block chaining message authentication code, *Journal of Computer and System Sciences*. **6**(3), 362–399, (2000).
38. DOE. Study of security attributes of smart grid systems — current cyber security issues, (April 2009). http://www.inl.gov/scada/publications/d/securing_the_smart_grid_current_issues.pdf.
39. M. Burmester, Y. Desmedt, and Y. Wang. Using approximation hardness to achieve dependable computation. In *RANDOM*, pp. 172–186, (1998).
40. Z. Zhao, Z. Dong, and Y. Wang, Security analysis of a password-based authentication protocol proposed to ieee 1363, *Theoretical Computer Science*. **352**, 280–287, (2006).