sSCADA: Securing SCADA Infrastructure Communications

Yongge Wang and Bei-Tseng Chu
Dept. of SIS, UNC Charlotte,
9201 University City Blvd, Charlotte, NC 28223,

August 5, 2004

Abstract

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 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. This paper presents a suite of security protocols optimized for SCADA/DCS systems which include: point-to-point secure channels, authenticated broadcast channels, authenticated emergency channels, and revised authenticated emergency channels. These protocols are designed to address the specific challenges that SCADA systems have.

1 Introduction

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. Without a secure SCADA system it is impossible to protect the nation's critical infrastructures. Indeed, the recent GAO report [15] shows that designing secure SCADA systems has the highest priority in protecting the nation's critical infrastructures.

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.

In a typical SCADA system [12], 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 large (e.g., large 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. Existing communication links between the control center and remote sites operate at very low speeds (could be on an order of 300bps to 9600bps). Figure 1 describes a simple SCADA system.

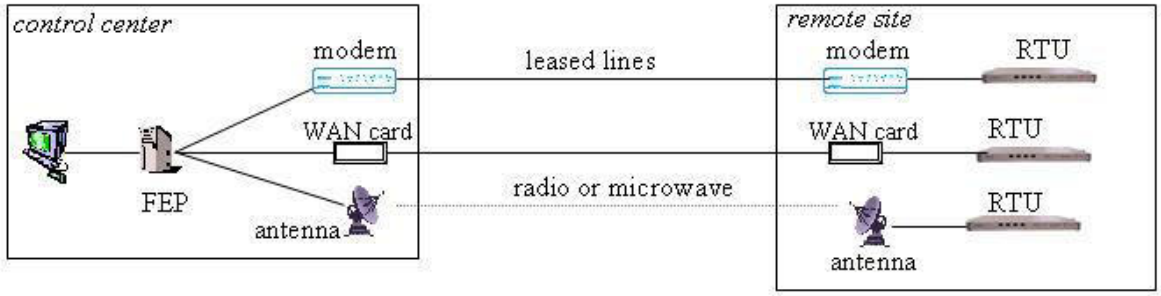

Figure 1: A simple SCADA system

In practice, more complicated SCADA system configurations exist. Figure 2 lists three typical SCADA system configurations (see, e.g., [4]).

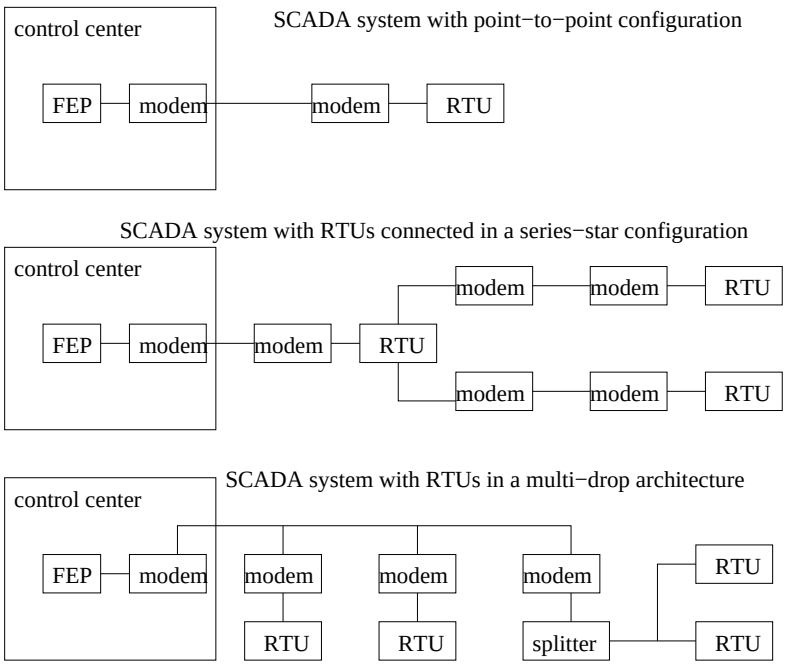

Figure 2: Typical SCADA system configurations

2 Threats to SCADA systems

An actual attack on SCADA systems was reported in [1]. In this attack, an Australian man hacked into the Maroochy Shire, Queensland computerized waste management system and caused millions of liters 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.

SCADA systems were not designed with public access in mind, they typically lack even rudimentary security. However, with the advent of technology 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 [3]. Similarly, frequency hopping spread spectrum radio and other wireless communication mechanisms frequently used to control remote terminal units (RTU) can be compromised as well. According to GAO's report [15], 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. For example, Sean Gorman's dissertation (see, e.g., [25]) 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. In addition, 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 [15] shows 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 SCADA remote communication links. In particular, we design new security technologies to secure SCADA systems.

3 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" [2], "American Gas Association (AGA) SCADA Encryption Committee" [6] has been trying to identify the functions and requirements for authenticating and encrypting SCADA communication links. Their proposal [4] 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 (see [4]):

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

This paper designs efficient cryptographic mechanisms to address these challenges and to build cryptographic modules as recommended in [4]. These mechanisms can be used to build plug-in devices (called sSCADA) 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.

The trust requirements in our security protocol design is as follows. RTU devices are deployed in untrusted environments and individual remote devices could be controlled by adversaries. The communication links are not secure but messages (maybe modified or re-ordered) could be delivered to the destination with certain probability. In another word, complete denial of service attacks (e.g., jamming) on the communication links are not addressed in our protocol. Compromising the control center in a SCADA system will make the entire system useless. Thus we assume that control centers are trusted in our protocol.

4 sSCADA protocol suite

The sSCADA protocol suite 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.

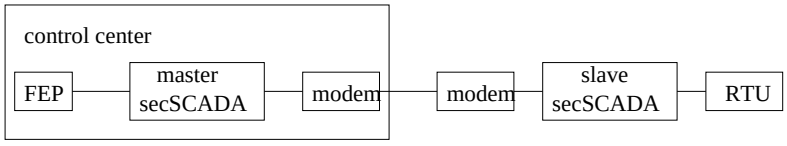

Figure 3: sSCADA with point-to-point SCADA configuration

4.1 Point-to-point secure channels

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 [17] 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 counter mode (CTR). 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. A similar design has been proposed in the sensor network communication protocol SNEP [22].

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 CTR 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 CTR 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 share 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 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.

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

[8, 20], CBC-MAC [19], and others. When party A wants to send a message m to party B securely, A computes the ciphertext $c = \mathcal{E}(C_A, K_{AB}, m)$ and message authenticator $mac = MAC(K'_{AB}, C_A||c)$, where $\mathcal{E}(C_A, K_{AB}, m)$ denotes the encryption of 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:

$$A \rightarrow B: \quad c,\ mac$$

When B receives the above packets, B decrypts c and verifies the message authenticator mac. There are several implementation issues on how to deliver the message to the target (e.g., RTU). For example, we give two cases in the following.

- B receives the entire package and verifies the message authenticator. During the verification procedure, the authenticity of the counter is verified implicitly. If the verification fails, the reason could be that the counter C_A is not synchronized. Thus B may try several possible counters until the authenticator is verified. B uses the verified counter and the corresponding key to decrypt the message and deliver the resulting message to the target. 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 disadvantage for this implementation is that these operations introduce additional delay which may interfere with SCADA protocols.

- In order to avoid delays introduced by cryptographic operations, 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 [5]. The disadvantage for this implementation is that B has no chance to try for different counters C_A. Thus it is not as robust as the previous implementation. However, if the chance that the counters are synchronized is significantly high, then this implementation is better.

- Another potential solution is to add two bytes of special format string to the plaintext message (two bytes bandwidth is normally available in SCADA systems) before encryption. When B receives a ciphertext, it decrypts it with the available counter value. If the special prefix is correct, then B has confidence that the counter value is synchronized. Otherwise, B tries other counter values to decrypt the correct prefix. After the counter value is determined, B continues the decryption and delivers the decrypted message to the target except the last byte. The last byte message is delivered only if the authenticator is verified.

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 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 a 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.

When necessary, it is also possible to configure sSCADA devices to do encryption only. That is, each time when A wants to send a message m to B, it computes the ciphertext $c = \mathcal{E}(C_A, K_{AB}, M)$ and sends c to B. The advantage of this encryption-only mode is that mac is not sent over the channel thus avoiding message overhead. This may be important in several SCADA applications since there is no additional bandwidth for the message authenticator at all. Indeed, weak authentication and message freshness are also achieved in the encryption-only mode if the counters are synchronized. B uses the synchronized counter C_A to decrypt the ciphertext c. If the message is generated by the adversary without the knowledge of the secret key or is a replayed ciphertext, then the decrypted message is generally not meaningful and the CRC code verification in the SCADA protocol will fail. Thus the SCADA protocol at the target will discard this message.

It is straightforward to show that our point-to-point secure channels provide data authentication, data integrity, data confidentiality, and weak data freshness (that is, messages arrive at the destination in the same order that was sent from the source).

4.2 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:

$$\begin{array}{ll} A \rightarrow B: & N_A \\ B \rightarrow A: & C_B,\ MAC(K'_{BA}, N_A||C_B) \end{array}$$

This counter synchronization protocol is analogous to that in [22].

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.

4.3 Authenticated broadcast channels

Encryption and authentication alone are not sufficient for SCADA applications. For example, it is not acceptable to authenticate a message individually in an emergent shutdown when timely responses from the RTU's are critical. In order to support authenticated broadcast, we use one way key chains. This channel can be used to establish other channels such as authenticated emergency channels (see next section).

Typical authenticated broadcast channels require asymmetric cryptographic techniques, otherwise any compromised receiver could forge messages from the sender. Cheung [13] proposed a symmetric cryptography based source authentication technique in the context of authenticating communication among routers. Cheung's technique is based on delayed disclosure of keys by the sender. Later, it was used in the Guy Fawkes protocol [7] for interactive unicast communication, and in [9, 10, 11, 23, 24] for streamed data multicast. Recently, Perrig, Szewczyk, Tygar, Wen, and Culler adapted delayed key disclosure based TESLA protocols [23, 24] to sensor networks for sensor broadcast authentication (the new adapted protocol is called μTESLA). One-way key chains used in these protocols are analogous to the one-way key chains introduced by Lamport [21] and the S/KEY authentication scheme [18].

In the following, we briefly describe the authenticated broadcast scheme for SCADA systems. At the sender (normally the master sSCADA device or a computer connected to it) set up time, the sender generates

a one-way key chain in the setup phase. In order to generate a one-way key chain of length n, the sender chooses a random key $\mathcal{K}_n$ first, then it applies the pseudorandom function $\mathcal{H}$ repeatedly to $\mathcal{K}_n$ to generate the remaining keys. In particular, for each $i < n$, $\mathcal{K}_i = \mathcal{H}(\mathcal{K}_{i+1})$.

For the purpose of broadcast authentication, the sender splits the time into even intervals I_i. The duration of each time interval is denoted as δ (e.g., $\delta = 5$ seconds or 5 minutes or even 2 hours), and the starting time of the interval I_i is denoted as t_i. In another word, $t_i = t_0 + i\delta$. At time t_i, the sender broadcasts the key $\mathcal{K}_i$. Any device that has an authentic copy of key $\mathcal{K}_{i-1}$ can verify the authenticity of the key $\mathcal{K}_i$ by checking whether $\mathcal{K}_{i-1} = \mathcal{H}(\mathcal{K}_i)$. Indeed, any device that has an authentic copy of some key $\mathcal{K}_v$ ($v < i$) can verify the authenticity of key $\mathcal{K}_i$ since $\mathcal{K}_v = \mathcal{H}^{(i-v)}(\mathcal{K}_i)$.

Let d (a unit of time intervals) be the key disclosure delay factor. The value of d is application dependent and could be configured at deployment time or after deployment. After d is fixed, the sender will use keying materials derived from key $\mathcal{K}_{i+d}$ to authenticate broadcast messages during the time interval I_i. Thus the message being broadcast during time interval I_i could be verified by the receiver during the time interval I_{i+d} after the sender broadcasts $\mathcal{K}_{i+d}$ at time t_{i+d}. It is easy to see that in order to achieve authenticity, the sender and the receiver need to be loosely time synchronized. Otherwise, if the receiver time is slower than the sender's time, an attacker can use published keys to impersonate the sender to the receiver. Typically the key disclosure delay should be greater than any reasonable round trip time between the sender and the receiver. If the sender does not broadcast data frequently, the key disclosure delay may be significantly larger. For example, $d\delta$ could take the value of several hours for some SCADA systems.

If a receiver (typically a slave sSCADA device) is deployed at some time during the interval I_i, the sender needs to bootstrap key $\mathcal{K}_i$ on the one-way key chain to the receiver. The sender also needs to bootstrap the key disclosure schedule which includes the starting time t_i of the time interval I_i, the key disclosure delay factor d, and the duration δ of each time interval. All these information could be bootstrapped to the receiver using the point-to-point secure channel that we have designed in the previous section or using other channels such as manual input. During a time interval I_j ($j > i$), the receiver receives the broadcast key $\mathcal{K}_j$ from the sender and verifies whether $\mathcal{K}_{j-1} = \mathcal{H}(\mathcal{K}_j)$. If the verification is successful, the receiver updates its key on the one-way key chain. If the receiver does not receive the broadcast key during the time interval I_j (either due to packet loss or due to active denial of service attacks such as jamming attacks), it can update its key in the next time interval I_{j+1}.

When a receiver gets a packet from the sender, it first checks whether the key used for the packet authentication has been revealed. If the answer is yes, then the attacker knows the key also and the packet could be a forged one. Thus the receiver needs to discard the packet. If the key have not been revealed yet, the receiver puts the packet in the buffer and checks the authenticity of the packet when the corresponding key is revealed. As stated above, if the sender and the receiver agree on the key disclosure schedule and the time is loosely synchronized, then message authenticity is guaranteed. However, the protocol does not provide non-repudiation, that is, the receiver cannot convince a third party that the message was from the claimed sender.

If we assume that the time between the sender and the receiver are loosely synchronized and the pseudorandom function $\mathcal{H}(\cdot)$ and the message authentication code (MAC) are secure, then an analogous proof as in [24] could be used to show that the above authenticated broadcast channel is secure. Note that we say that a *pseudorandom function* $\mathcal{H}(\cdot)$ is secure if the function family $f_k(x) = \mathcal{H}(k, x)$ is a pseudorandom function family in the sense of [16] when k is chosen randomly. That is, a function family $\{f_k(\cdot)\}$ is pseudorandom if the adversary with polynomially bounded resources cannot distinguish between a random chosen function from $\{f_k(\cdot)\}$ and a totally random function with non-negligible probability. We say that a message authentication scheme MAC is secure if a polynomially bounded adversary will not succeed with non-negligible probability in the following game. A random l-bits key k are chosen by the user. The adversary chooses messages $m_1, \ldots, m_t$ and the user generates the MAC codes on these messages using the key k. The adversary succeeds if she could then generate a MAC code on a different message $m' \neq m_1, \ldots, m_t$.

Though the time synchronization between the sender and the receiver plays an important role in the security of the protocol, they do not need to have 100% accurate clocks. If their clocks are sufficiently accurate, then time synchronization protocol could be designed to synchronize their clocks to meet the security requirements. The time synchronization protocols could be based on the point-to-point secure channels discussed in the previous section.

In the above authenticated broadcast protocol, the receiver cannot verify the authenticity of the message immediately since it needs to wait for the disclosure of the key after a time period of $d\delta$. This is not acceptable for some broadcast messages such as an emergency shutdown. In order to overcome this challenge, the sender may reveal the key used for emergency messages immediately or shortly after the message broadcast. This will open the door for an adversary to modify the emergency messages. For example, if the message passes through a node D before it reaches a node C, D can discard the message and create a different emergency message and forward it to C. In another case, an attacker may jam the target C during the emergency broadcast period and sends C a different emergency message (authenticated using the revealed key for the emergency message) later. If these attacks are not practical, then it is OK to use this immediate-key-disclosure protocol for emergency broadcast. Otherwise, we need to use the emergency channels that we will discuss in the next sections.

4.4 Authenticated emergency channels

As stated in the previous section, authenticated emergency channels are needed for many SCADA systems. In this section we design such a channel based on the authenticated broadcast channel. Generally there are limited number of emergency messages. Assume that these emergency messages are $e_1, \ldots, e_u$. Without loss of generality, we may assume that $e_i = i$ for $i \leq u$. Before the sender could authentically broadcast these messages, it needs to carry out a commitment protocol.

Let v be a fixed number. During the message commitment procedure, the sender chooses v random numbers $N_1^i, \ldots, N_v^i$ for each $i \leq u$. It then computes $r_{i,j} = \mathcal{H}(e_i||N_j^i)$ for all $i \leq u$ and $j \leq v$. Using the authenticated broadcast channel, the sender broadcasts the commitments $\{r_{i,j} : i \leq u \text{ and } j \leq v\}$ to all receivers. Receivers store these commitments in their memory space.

Each time when the sender wants to broadcast the message e_i to receivers emergently, it chooses a random unused $j \leq v$, and broadcasts (e_i, j, N_j^i) to all receivers. The receiver verifies that $r_{i,j} = \mathcal{H}(e_i||N_j^i)$. If the verification is successful, it knows that the message e_i comes from the sender and delivers it to the target. At the same time, the receiver deletes the commitment $r_{i,j}$ from its memory space.

Note that after each message commitment procedure, the sender could broadcast each message at most v times. Thus the sender may decide to initiate the message commitment protocol when any one of these messages has been broadcast sufficiently many times (e.g., $v - 1$ times). Each time when the message commitment protocol is initiated, both the sender and the receiver should delete all previous commitments from their memory space.

The security of the emergency channel could be proved formally under the assumption that the pseudorandom function $\mathcal{H}(\cdot)$ is a secure one-way function. That is, for any given y with appropriate length, one cannot find an x such that $\mathcal{H}(x) = y$ with non-negligible probability.

Theorem 4.1 *Assume that the authenticated broadcast channel is secure and the pseudorandom function $\mathcal{H}(\cdot)$ is a secure one-way function. Then the authenticity of messages that receivers accept from the emergency channel is guaranteed.*

Sketch of Proof. Assume for a contradiction that the authenticity of the emergency protocol is broken. That is, there is an adversary $\mathcal{A}$ who controls communication links and manages to deliver a message m to the receiver such that the sender has not sent the message but the receiver accepts the message. We show in the following that then $\mathcal{H}(\cdot)$ is not a secure one-way function. Specifically, let t be the total number of messages

that the sender can broadcast in the emergency channel with one commitment $\{r_{i,j}\}$, and $y_1, \ldots, y_t$ be t randomly chosen strings with appropriate lengths (i.e., they are potential outputs of $\mathcal{H}$). We will construct an algorithm $\mathcal{P}$ that uses $\mathcal{A}$ to compute a pre-image $x = \mathcal{H}^{-1}(y_i)$ of some string y_i with non-negligible probability.

Since the broadcast channel is secure, we can always assume that the commitment $\{r_{i,j}\}$ that the receivers accept are authentic. The algorithm $\mathcal{P}$ works by running $\mathcal{A}$ as follows. Essentially, $\mathcal{P}$ simulates an authenticated broadcast channel for $\mathcal{A}$ with a sender A and a receiver B.

1. $\mathcal{P}$ chooses a random number $l \leq t$.
2. $\mathcal{P}$ computes a commitment $\{r_{i,j}\}$ as specified in the emergency broadcast protocol. $\mathcal{P}$ picks $t - l + 1$ random values from the commitment $\{r_{i,j}\}$ and replace them with $y_l, y_{l+1}, \ldots, y_t$.
3. $\mathcal{P}$ runs the sender's algorithm to authentically broadcast the modified commitment to B.
4. For the first $l - 1$ emergency messages, $\mathcal{P}$ runs the sender's algorithm of the emergency broadcast protocol with no modification to broadcast the pre-images of the $l - 1$ unmodified commitments.
5. $\mathcal{P}$ then waits for $\mathcal{A}$ to deliver a fake message x' that B accepts as an authentic emergency broadcast. $\mathcal{P}$ outputs x' as one of the pre-images of $y_l, \ldots, y_t$.

We briefly argue that $\mathcal{P}$ outputs the pre-image of one of the strings from $y_1, \ldots, y_t$ with non-negligible probability. Since $\mathcal{A}$ succeeds with non-negligible probability in convincing the receiver to accept a fake message, it must deliver this message as the l-th message for some $l \leq t$ in the authenticated emergency channel. Thus for this l, the algorithm $\mathcal{P}$ outputs a pre-image for one of the given strings with non-negligible probability. Q.E.D.

Theorem 4.1 shows that messages received in the emergency channel are authentic. However, it does not show whether these messages are fresh. Indeed, when the sender broadcasts an emergency message at the time t, the adversary may launch a denial of service attack against the receiver or just does not deliver the message to the receiver. Thus the receiver will not be able to delete the commitment of this message from its memory space. Later at time t', the adversary delivers this message to the receiver and the receiver accepts it. In our emergency channel, there is no way to avoid this kind of delayed message attacks. Thus when message freshness is important, one may use the revised authenticated emergency broadcast channel that we will discuss in the next section.

4.5 Revised authenticated emergency channel

There are basically two ways to guarantee the freshness of a received message. The first one is to use public key cryptography together with time-stamps. The second solution is to let the receiver send a nonce to the sender first and the sender authenticates the message together with the nonce. As we have mentioned earlier, public key cryptography is too expensive to be deployed in SCADA systems. For the second solution, the delays introduced in nonce submission process are generally not acceptable in an emergent situation. In this section, we introduce a revised emergency broadcast protocol, which provides weak freshness of received messages.

Let the u emergency messages be $e_1, \ldots, e_u$. Similar to the previous protocol, the sender needs to carry out a commitment protocol before the authenticated emergency broadcast. In the revised protocol, the sender chooses v random numbers $N_1^i, \ldots, N_v^i$ and v expiration time points $T_1^i < T_2^i < \ldots < T_v^i$ for each $i \leq u$. It then computes $r_{i,j} = \mathcal{H}(e_i || N_j^i || T_j^i)$ for $i \leq u$ and $j \leq v$. Using the authenticated broadcast channel, the sender broadcasts the commitments $\{r_{i,j} : i \leq u \text{ and } j \leq v\}$ to all receivers. Receivers store these commitments in their memory space. The functionality of expiration time points in the revised protocol is

to guarantee that the commitment $r_{i,j}$ for the message e_i expires at the time T_j^i. In another word, when the receiver receives (e_i, N_j^i, T_j^i), it will accepts the message only if the current clock time of the receiver is earlier than T_j^i.

If the sender wants to send the message e_i to receivers at time t, it chooses a random unused $j \leq v$ such that $t < T_j^i$, the estimated transmission time from the sender to receiver is less than $T_j^i - t$, and T_j^i is the earliest time in the commitments that satisfies these conditions. Then the sender broadcasts (e_i, j, N_j^i, T_j^i) to all receivers. The receiver verifies that $r_{i,j} = \mathcal{H}(e_i||N_j^i||T_j^i)$ and the current clock time of the receiver is earlier than T_j^i. If the verification is successful, it knows that the message e_i comes from the sender and delivers it to the target. At the same time it deletes the commitment $r_{i,j}$ from its memory space. Otherwise, the receiver discards the message.

The implementation of the revised emergency broadcast protocol has the flexibility to choose the gaps between expiration time points T_j^is for each $i \leq u$. The smaller the gap, the better the freshness property. However, smaller gaps between T_j^is add additional overhead on the communication links. It is also possible, for different messages e_i, one chooses different values v. For example, for more frequently broadcast message, the value of v should be larger. It is also important to guarantee that the commitment is always sufficient and when only a few commitments are unused, the sender should initiate a procedure for a new commitment.

The security of the revised emergency broadcast protocol can be proved similarly as in Theorem 4.1. It is still possible for an adversary to delay an emergency message (e_i, j, N_j^i, T_j^i) broadcast by the sender during the time period $[T_{j-1}^i, T_j^i]$ until T_j^i. However, she cannot delay the message to some time points after T_j^i. In another word, weak freshness of received messages are guaranteed in the revised authenticated emergency channel.

5 Conclusion

In this paper, we introduced a security protocol suite that addresses the challenges of SCADA communication environments. The sSCADA devices are under the process of development at the moment. After these devices are built, they will be tested in the real SCADA system environments. However, our theoretical analysis shows that the sSCADA devices can solve all these challenges and will not interfere with SCADA protocols.

References

[1] http://www.theregister.co.uk/content/4/22579.html

[2] http://www.mel.nist.gov/proj/cip.htm

[3] http://www.tscm.com/outsideplant.html

[4] AGA Report No. 12. Cryptographic Protection of SCADA Communications: General Recommendations. Draft 2, February 5, 2004. http://www.gtiservices.org/security/AGA12Draft2r21.pdf

[5] AGA report No. 12-1. Cryptographic Protection of SCADA Communications. Draft 1, March 24, 2003. http://www.gtiservices.org/security/

[6] American Gas Association Security Group. http://www.gtiservices.org/security/

[7] R. Anderson, F. Bergadano, B. Crispo, J. Lee, C. Manifavas, and R. Needham. A new family of authentication protocols. *Operating Systems Review*, **32**(4):9–20, October, 1998.

[8] M. Bellare, R. Canetti, and H. Krawczyk. message authentication using hash functions–the HMAC construction. *RSA Laboratories CryptoBytes* **2**(1), Spring, 1996.

[9] F. Bergadano, D. Cavagnino, and B. Crispo. Chained stream authentication. In: *Selected Areas in Cryptography*, Waterloo, Canada, 2000.

[10] F. Bergadano, D. Cavagnino, and B. Crispo. Individual single source authentication on the mbone. In: *ICME 2000*, August 2000.

[11] B. Briscoe. FLAMeS: Fast, Loss-Tolerant Authentication of Multicast streams. Technical report 2000. http://www.labs.bt.com/people/briscorj/papers.html

[12] T. Cegrell. *Power System Control Technology*, Prentice-Hall International (UK) Ltd. 1986.

[13] S. Cheung. An efficient message authentication scheme for link state routing. In: *13th Annual Computer Security Applications Conference*, 1997.

[14] DOE. 21 steps to to improve cyber security of SCADA networks. http://www.ea.doe.gov/pdfs/21stepsbooklet.pdf

[15] GAO-04-628T. 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

[16] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. of the ACM* **33**(4):792–807, 1987.

[17] S. Goldwasser and S. Michali. Probabilistic encryption. *Journal of Computer and System Sciences* **28**:270-299.

[18] N. Haller. The S/KEY one-time password system. IETF RFC 1760, February 1995.

[19] NIST. DES model of operation, FIPS Publication 81 (FIPS PUB 81), 1981.

[20] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: keyed-hashing for message authentication. Internet RFC 2104, February 1997.

[21] L. Lamport. Password authentication with insecure communication. *Commun. ACM*, **24**(11), 1981.

[22] A. Perrig, R. Szewczyk, J. Tygar, V. Wen, and D. Culler. SPINS: Security protocols for sensor networks. *Wireless Networks* **8**:521–534, 2002.

[23] A. Perrig, R. Canetti, D. Song, and J. Tygar. Efficient and secure source authentication for multicast. In: *Network and Distributed System Security Symposium, NDSS'01*, 2001.

[24] A. Perrig, R. Canetti, J. Tygar, and D. Song. Efficient authentication and signing of multicast streams over lossy channels. In: *IEEE Symposium on Security and Privacy*, 2000.

[25] Washington Post. Dissertation could be security threat. http://www.washingtonpost.com/ac2/wp-dyn/A23689-2003Jul7.

[26] 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*, pages 263-277, LNCS 3809, Springer 2004.