Chapter 10

SECURITY ENHANCEMENTS FOR DISTRIBUTED CONTROL SYSTEMS

Jeffrey Hieb, James Graham and Sandip Patel

Abstract Security enhancements for distributed control systems (DCSs) must be sensitive to operational issues, especially availability. This paper presents three security enhancements for DCSs that satisfy this requirement: end-to-end security for DCS protocol communications, role-based authorization to control access to devices and prevent unauthorized changes to operational parameters, and reduced operating system kernels for enhanced device security. The security enhancements have been implemented on a laboratory-scale testbed utilizing the DNP3 protocol, which is widely used in electrical power distribution systems. The test results show that the performance penalty for implementing the security enhancements is modest, and that the implemented mechanisms do not interfere with plant operations.

Keywords: DNP3, secure communication, role-based authorization, RTU security

1. Introduction

Distributed control systems (DCSs) are networks of computer systems used for measurement and control of physical systems. They play a vital role in the operation of geographically-distributed critical infrastructures such as gas, water and electrical power distribution and the railroad transportation system. DCSs are also integral to chemical plants, refineries and water treatment facilities. The 1997 report of the President's Commission on Critical Infrastructure Protection [30] and the 1998 Presidential Decision Directive 63 [9] stressed the vulnerabilities of DCSs to cyber attacks. For years, DCSs have been relatively secure because of their isolation and obscurity, but recent data indicates that cyber attacks against these systems are on the rise [8].

Industrial control systems present unique security challenges. DCSs are widely-dispersed, complex, real-time systems that provide instrumentation and telemetry for real-world processes. Delays or lack of availability that might be acceptable in traditional information technology environments are unacceptable

in DCSs. Consequently, securing DCSs requires the design and implementation of real-time, high-speed and low-overhead solutions that do not interfere with industrial plant operations.

This paper presents three security enhancements to DCSs that satisfy these requirements: end-to-end security for DCS protocol communications, role-based authorization to control access to devices and prevent unauthorized changes to operational parameters, and reduced operating system kernels for enhanced device security. The security enhancements have been implemented and evaluated on a laboratory-scale testbed utilizing the DNP3 protocol, which is widely used in electrical power distribution systems.

2. DCS Security

Early control systems used a combination of knobs, dials and lights mounted on custom-built control panels. Communication with process machinery and field equipment was achieved using analog control signals carried by dedicated cables that connected the process control panels to field equipment [7]. Securing these systems was simply a matter of locking the door to the control room. Eventually, control systems began to use digital signals on serial lines based on the RS-232, RS-422 and RS-485 standards. This meant that, while networks were still relatively isolated, there was a consolidation of communication channels and communication standards. Distributed control systems (DCSs) of this era were still special-purpose stand-alone systems that were not intended to be connected to other systems [22]. They used vendor-developed proprietary protocols for communications between master terminal units (MTUs) and remote terminal units (RTUs). Due to the low fidelity and limited channel capacity of early serial communications, these protocols supported only the minimal functionality needed to achieve reliable scanning and control of remote devices [25].

Modern DCSs have been influenced by the successful use of open standards and commodity systems in information technology that have realized significant cost reductions through competition and economies of scale. This has led to the creation of modern DCS networks that are characterized by open architectures and open communication standards such as DNP3, MODBUS and IEC 60870. But the resulting network convergence has exposed DCSs to significant security threats [11, 27, 28]. The lack of authentication in DCS protocols makes communications vulnerable to spoofing, modification and replay attacks. Furthermore, the use of commercial off-the-shelf (COTS) hardware and software, especially commercial operating systems, in DCS devices makes them vulnerable to common cyber attacks. Attacks on DCSs can have serious consequences, including loss of service to utility customers, financial loss to service providers due to damaged equipment and corruption of metering information, environmental damage, even the loss of human life.

Mitigation of the risks posed by cyber attacks on DCSs has received increasing attention over the past few years. Several articles in the literature describe best practices for securing control networks [1, 11, 26, 28]. In general, security vulnerabilities are mitigated using well-established network security techniques

(e.g., network segmentation, access control, VPNs and firewalls) along with standard IT security policies and practices (e.g., strong passwords). Formal standards and guidelines for control network security can be found in documents from NIST [33] and ISA [17, 18].

There has been some work on developing DCS-specific security solutions. The American Gas Association (AGA) has been working to develop a "bump in the wire" (in-line) cryptographic solution for securing point-to-point serial communications in DCS networks [4, 5, 36]. Another bump in the wire solution using COTS equipment is described in [31]. Other efforts have produced a process-control-specific intrusion detection system [24], a DoS-deterrent technique for IP-based DCS devices [6], and an improved authentication and authorization technique for maintaining port access to field devices [35].

3. DCS Security Enhancements

This section discusses three DCS security enhancements that go beyond network perimeter defenses. The enhancements are: (i) securing DCS communications, (ii) restricting operations on RTUs, and (iii) hardening RTU operating systems. In addition, a security architecture for RTUs is described.

3.1 Security-Enhanced DNP3 Communications

Two techniques for enhancing security in DNP3 communications were presented in [15]. The first uses digital signatures to verify sender identity and message integrity. The second uses a challenge-response approach to allow either party to spontaneously authenticate the sender and verify the integrity of the most recently received message. The enhancements were formally verified using OFMC and SPEAR II, and were found not to contain flaws [29].

Authentication using Digital Signatures Authentication via digital signatures is implemented by appending an authentication fragment (AF) to each DNP3 message. The AF contains an encrypted hash digest of the message concatenated with a timestamp and nonce. The timestamp is used by the receiver to verify that the time of reception does not vary from the time of transmission by a pre-specified amount. The digest is encrypted using the sender's private key, but the message itself is not encrypted to reduce processing time. The receiver decrypts the hash digest using the sender's public key and compares it with the hash digest it calculates independently. If the decrypted AF matches the computed hash digest of the received message (excluding the AF), the receiver concludes that the message is unaltered and comes from an authentic source.

Authentication via Challenge-Response Authentication using challenge-response permits the verification of the identity of the communicating party and the integrity of the most recent message. The challenge-response mechanism requires that all parties possess a shared secret. Either device

(master or field unit) can initiate the challenge. The mechanism involves the following steps:

1. After the link is established, the authenticating device sends a random "challenge" message to the other device.

2. The other device responds with a value calculated using a one-way hash function. The hash stream contains the shared secret so that only a valid device can compute the correct hash value.

3. The challenger checks the response against the hash value it computes. If the values match, the DNP3 operation proceeds; otherwise the connection is terminated.

4. The authenticator sends new challenges to the other device at random intervals and repeats Steps 1–3 above.

Typically, a device would issue a challenge when an initial connection is created to prevent any further communication until the other device is authenticated. However, it is important that devices also issue challenges periodically to protect against man-in-the-middle attacks. For example, a device should issue a challenge immediately upon receiving a request to perform a critical operation, but before taking any action. To protect against replay attacks, the challenge message should contain data that changes randomly each time a challenge is issued. As usual, the responder must perform the cryptographic algorithm specified in the challenge message to produce the correct response.

3.2 RTU Authorization Model

In addition to external attacks, RTUs also face insider threats. While insider threats can never be completely mitigated, restricting users to authorized operations can limit the threat and constrain potential damage. This section describes an authorization model for controlling operations in a security-hardened RTU.

RTUs are typically connected to sensors and actuators. Central to RTU operation is a set of data values referred to as "points." These data values are digital representations of the telemetry and control provided by an RTU. "Status points" represent values read from a sensor (e.g., temperature); "command points" dictate the behavior of connected actuators; and "set points" influence local control algorithms. A security-hardened RTU should limit an individual user of a DCS to a set of authorized points and operations on those points. The possible operations on points for standard DCS communications (read, select and operate) are described in [13].

Access control to RTU points employs a role-based access control (RBAC) model [12] with an added constraint for expressing restrictions on permissions granted to roles based on the type of point. The subjects of the model are DCS users. Table 1 presents the access control model, including its key elements and functions.

Table 1. RTU access control model.

Function	**Arguments**	**Preconditions**	**Postconditions**
create_session	user, session	user $\in SU$; session $\notin S$	session $\in S$; session_role(session) = r | r $\in SR \wedge$ (user, r) $\in SUA$; user_session(session) = user
delete_session	session	session $\in S$	session $\notin S$; session_role(session) = null; user_session(session) = null
check_access	session, op, p, result	session $\in S$; op $\in SOP$; p $\in P$	result = (op, p, session_role(session)) $\in PA$
add_user	user	user $\notin SU$	user $\in SU$; $\neg\exists r \in SR$| (user, r) $\in SUA$
delete_user	user	user $\in SU$	user $\notin SU$; $\neg\exists r \in SR$| (user, r) $\in SUA$
assign_user	user, role	user $\in SU$; role $\in SR$; $\neg\exists$ r $\in SR$| (user, r) $\in SUA$	(user, role) $\in SUA$
deassign_user	user, role	user $\in SU$; role $\in SR$; (user, role) $\in SUA$	(user, role) $\notin SUA$
assign_role	role, op, obj	role $\in SR$; (op, obj) $\in PER$; (role, type(op)) $\in RT$	((op, obj), role) $\in PA$
deassign_role	role, op, obj	role $\in SR$; (op, obj) $\in PER$; ((ob, obj), role) $\in PA$	((op, obj), role) $\notin PA$

SU: Set of DCS users; SR: Set of DCS roles; S: Set of sessions;

SUA: Many to one mapping of users to roles ($SU \times SR$);

SOP: Set of DCS operations and administrative operations;

P: Set of RTU points; PER: Set of permissions ($SOP \times P$); PT: Set of point types;

PA: Many to many mapping of permissions to roles ($PER \times SR$);

PTA: Many to one mapping of points to point types ($P \times PT$);

RT: Set of tuples $SR \times PT$ indicating which point types a role may operate;

user_session(s:S) $\rightarrow$ u:SU: Function mapping each session s_i to a single user;

session_role(s:S) $\rightarrow$ r:SR: Function mapping each session s_i to a role;

type(p:P) $\rightarrow$ pt:PT: Function mapping each point to a type.

The access control model includes both DAC and MAC components. Dynamic modifications are limited to the addition and deletion of users, assignment and de-assignment of users to roles, and assignment of permissions to roles. These operations are subject to some MAC constraints, which are defined by the relations SUA, RT and PA. SUA enforces the constraint that every user can be assigned only one role. RT defines the point types a particular role may act upon. PA enforces constraints related to permissions and roles, e.g., an administrator cannot perform any operation other than administrative tasks and no user can obtain all permissions. Other elements of the model are considered to be fixed for a particular RTU and are set when the RTU is configured.

3.3 Reduced Kernels for RTUs

System vulnerabilities introduced by COTS components, such as commercial operating systems, expose RTUs to common attacks that circumvent protocol and application layer security controls and allow attackers access to critical RTU resources. This section describes two reduced kernel approaches for providing a hardened operating system base for RTUs; in addition, it presents a high-level security architecture for RTUs.

Operating systems play a central role in security because they mediate all access to shared physical resources. The operating system kernel provides the lowest level of abstraction between the hardware and the rest of the system through the system call interface, and implements access control mechanisms to protect system objects and processes. In the case of RTUs, flaws and vulnerabilities in the operating system kernel and misconfigured security settings can allow malicious code to modify or interfere with other running applications (e.g., local control algorithms and DCS applications) or bypass security mechanisms and directly access the I/O ports that operate field equipment. In the following, we describe two minimal kernel approaches for creating a hardened RTU kernel.

As mentioned previously, clear economic advantages exist to using COTS operating systems in RTUs and other field devices. But today's commodity operating systems have large monolithic kernels and contain numerous known and unknown vulnerabilities that are inherited by RTUs. A simple and straightforward approach to address this problem is to minimize the COTS operating system to include only the components needed for RTU operations. Enhanced RTU security is achieved through reduced complexity and the elimination of vulnerabilities due to the exclusion of non-essential components.

The second approach involves the use of a microkernel architecture [20], i.e., a minimal kernel that implements only those services that cannot be implemented in user space. Microkernels have three minimal requirements: address space, inter-process communication and unique identifiers. The virtues of a microkernel include greater stability, reduced TCB and less time spent in kernel mode. The MILS initiative has developed a high-assurance, real-time architecture for embedded systems [3, 16]. The core of the MILS architecture is a

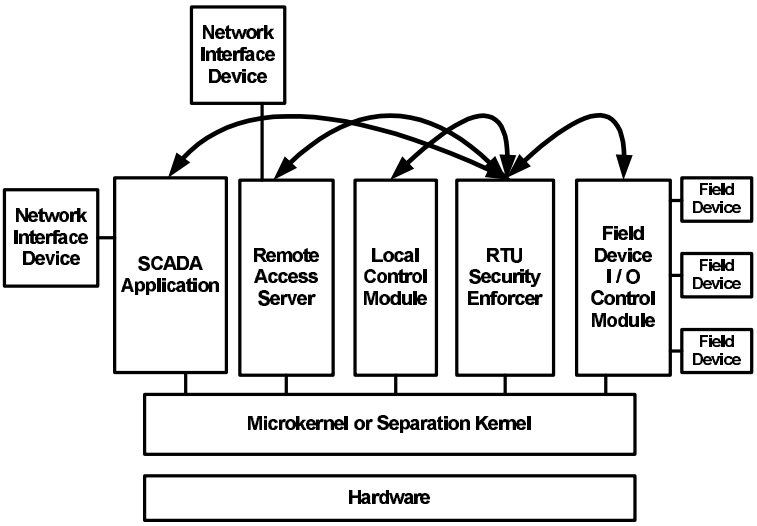

Figure 1. Microkernel-based RTU.

separation kernel, which is small enough (approximately 4,000 lines of code) to be formally verified. The separation kernel isolates processes and their resources into partitions. Processes running in different partitions can neither communicate nor interfere with processes in other partitions unless explicitly allowed by the kernel. MILS leverages the partitioning to allow security functions traditionally implemented in the operating system kernel to be moved into their own isolated partitions. These modules, which are part of the MILS middleware, are also small enough to be formally verified.

A hardened RTU can be created using a separation kernel or similar microkernel. The design places various RTU functional components in their own partitions or address spaces with well-defined communication paths (Figure 1). Digital and analog I/O modules can be placed in separate partitions and given exclusive access to the appropriate hardware. RTU applications that provide network services are placed in their own partitions as well. Finally, a security enforcement module is positioned between the partitions to provide mandatory enforcement of the RTU security policy.

3.4 Security Architecture for RTUs

The proposed security-enhanced RTU architecture builds on the microkernel concept of isolating system components and security functions in their own partitions. Figure 2 presents a high-level description of the security-enhanced RTU architecture. In the model, only an I/O controller has access to analog and digital I/O ports. Access to status points and command points is restricted by the access control enforcement and security functions modules, which provide a public interface for RTU services and share a private (trusted) communica-

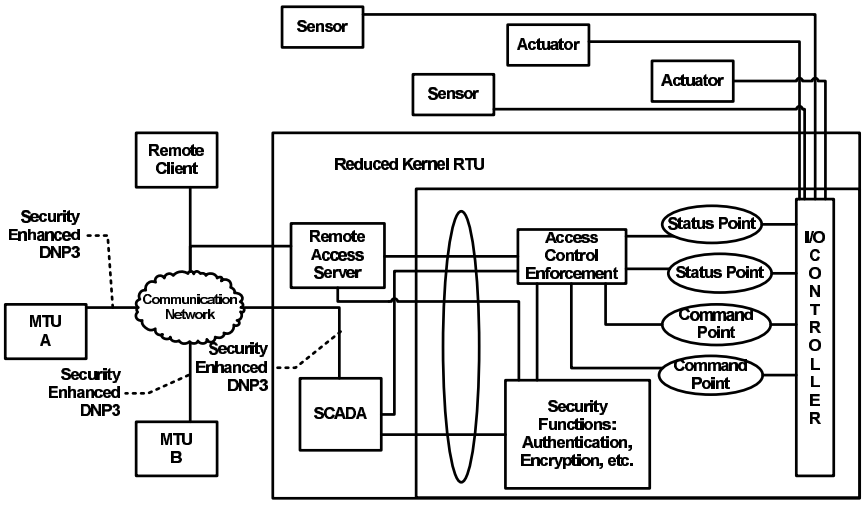

Figure 2. Security-enhanced RTU architecture.

tion interface for security-relevant information. All access to RTU points is via the access control enforcement module, where access control decisions are influenced by the access control policy and trusted security attributes obtained from protected and verified security functions.

4. Experimental Results

In our preliminary experiments, standard PCs were used to emulate RTUs and tests were conducted to measure the relative performance of the security enhancements. A more elaborate testbed is currently under development. The testbed incorporates a binary distillation column and a water-level control system. This testbed will also incorporate a hardened RTU prototype developed by modifying a commercially-available SIXNET RTU.

The DNP3 security enhancements involving authentication fragments and challenge-response authentication were tested in a simulated DCS environment. A minimal kernel RTU was created using LynxOS from LynuxWorks [21] that ran on a standard PC; this prototype also provided role-based access control to simulated device points. A MILS system or separation kernel was not available for testing.

4.1 Security-Enhanced DNP3 Communications

The authentication fragment (AF) and challenge-response enhancements were implemented on a DCS testbed [29] that simulated a subset of DNP3 MTU–RTU communications. SHA-256 was used as the hash function for the AF implementation. In the preliminary experiments, the hash was encrypted

Table 2. Performance of security-enhanced DNP3 communications.

	Total Time (ms)	**MTU (ms)**	**RTU) (ms)**
DNP3	325	4	66
DNP3 with AF and software encryption	2,146	340	1,168
DNP3 with AF and hardware encryption (est.)	764	22	104
DNP3 with challenge response	446	25	32

using AES-128; note, however, that an asymmetric cryptographic algorithm and PKI would be required for the complete implementation.

The challenge-response algorithm used a four-byte shared secret. The SHA-256 hashing algorithm was used by the MTU and RTU. The MTU was a 1.0 GHz Intel Pentium IV PC running Windows XP and web server software to provide an HMI. The RTU was a Windows-2000-based 350 MHz PC with 256 MB of RAM, which was connected to the DCS hardware.

The goal of the performance analysis was to assess the relative impact of the enhancements on communication latency. Table 2 shows the time requirements in milliseconds (ms) for processing an entire message, along with the time required by the MTU and RTU to process a message before sending a reply. The baseline values were provided by an implemented subset of DNP3 without security enhancements. As expected, encryption comes at a cost (Row 2 in Table 2). However, the performance can be improved significantly using hardware encryption (Row 3). A field-programmable gate array (FPGA) provides a low-cost, practical solution to the encryption/decryption needs of the authentication fragment model and provides throughput up to 18 Gbps [34]. Note that a conservative throughput of 10 Mbps was used to calculate the values in Table 2, assuming that a DCS network uses an in-line encryption device, which is considerably slower than other encryption devices.

4.2 RTU Authorization

Access control on the RTU was implemented as a middleware layer that had access to all the simulated device points and that provided an external interface for applications using IPC `msgsend` and `msgrecv` calls. Applications such as the DNP3 module retrieve points through IPC and use authentication credentials (initially userid and password) to establish a session for reading and writing points. The access control policy was stored in files accessible only to the enforcement module. The operation `check_permission(session, permission)` was used to apply the policy by searching for a matching permission assigned

to the role associated with the session. A DNP3 module was implemented to provide a DNP3 interface to the RTU. DNP3 over TCP/IP was used on the laboratory LAN. The RTU was configured with three users, each assigned a different role (`engineer`, `operator`, `monitor`). An MTU was implemented to interrogate the RTU by polling the RTU for each point and then writing the command points back to the RTU. Three timing measurements were collected while the MTU interrogated the RTU using different users: (i) time elapsed while the MTU waited on an RTU response, (ii) time elapsed between the DNP3 module receiving a message and sending a response, and (ii) time the DNP3 module spent blocked on IPC `msgrecv`, i.e., waiting on the access control module.

The interrogation of the MTU was initially performed with the access control call `check_permission()` disabled; the call was subsequently enabled to determine the relative performance impact. Without the RTU's role-based access control feature, the MTU experienced a mean response time of 0.45 ms and a worst case response time of 0.70 ms. The mean time taken by the RTU to process a DNP3 message was 71 μs and the DNP3 module spent 32 μs blocked waiting on the IPC. With role-based access control enforced, the mean response time experienced by the MTU was 0.70 ms, with a worst case response time of 1.56 ms. On the RTU, the mean time to process a DNP3 request was 198 μs, with 146 μs spent blocked waiting on the IPC. As expected, there is some performance impact, but the impact is small, an increase of just 0.25 ms on the average. Since most continuous polling techniques have built-in delays [32], a small increase in response time does not impact system stability and throughput. However, the addition of many users and permissions, manifested by a large number of points on the RTU, would lead to performance degradation; therefore, suitable modeling and optimization techniques will have to be investigated. Furthermore, actual DCS traffic is needed to conduct a more thorough analysis of the performance impact of the access control implementation.

4.3 Reduced Kernel RTU

A prototype reduced kernel RTU was developed on a standard PC using the real-time OS (RTOS) LynxOS from LynuxWorks [21]. The RTU had a total of ten simulated points, and the access control model described in Section 3.2 was also integrated into the prototype. The DNP3 security enhancements were developed in parallel so that authentication used a username and password with the assumption that future prototypes would use authentication schemes compatible with protocol enhancements. A subset of DNP3 was used for RTU–MTU communications, and was extended to include an authentication credentials request function `0xF7`, an authentication object (group `0x20`) comprising a username and password, and an internal indicator status flag to indicate if authentication failed or a session timed out. The MTU was implemented to interrogate the RTU by polling each device point then writing back to each device output point.

To create a reduced kernel RTU, all unnecessary device drivers (SCSI, IDE, USB, etc.) and support for NFS and IPv6 were removed from the kernel. The size of the standard kernel was approximately 1.4 MB; the reduced kernel was 906 KB, a reduction of approximately 36%. We believe that significant additional reductions can be achieved by conducting a fine-grained (and tedious) examination of kernel components to identify unused kernel libraries and routines and by modifying kernel parameters that affect kernel data structures. System binaries and libraries, which also make up the operating system, were reduced as well. This involved excluding binaries and libraries that were not needed for RTU operation. Of particular relevance are unneeded network services such as `finger` and RPC, which could be initiated inadvertently or maliciously to provide an attacker with additional vectors. The kernel and the system binaries together make up the boot image, which was reduced from 4.7 MB to 2.5 MB. We expect to reduce the boot image even further through reductions in kernel size and by conducting a detailed analysis of library dependencies.

5. Conclusions

DCSs are large distributed networks with a variety of architectural components; consequently, securing these systems requires security mechanisms to be embedded throughout their different components and layers. However, most DCS security strategies have focused on applying standard IT security technologies. In contrast, the security enhancements presented in this paper are designed specifically for DCSs. The enhancements, which include end-to-end security for DCS protocol communications, role-based authorization to control access to devices and prevent unauthorized changes to operational parameters, and reduced operating system kernels for enhanced device security, balance security and availability. The performance penalty for implementing the security enhancements is modest; simulation results demonstrate that they do not interfere with plant operations. Future research will concentrate on extending and refining the secure communication and access control strategies for use in large-scale industrial environments. Efforts will also be undertaken to harden RTU operating systems by reducing kernel size while embedding security within the kernel.

References

[1] J. Abshier, Ten principles for securing control systems, *Control*, vol. 18(10), pp. 77–81, 2005.

[2] J. Abshier and J. Weiss, Securing control systems: What you need to know, *Control*, vol. 17(2), pp. 43–48, 2004.

[3] J. Alves-Foss, C. Taylor and P. Oman, A multi-layered approach to security in high assurance systems, *Proceedings of the Thirty-Seventh Annual Hawaii International Conference on System Sciences*, 2004.

[4] American Gas Association, Cryptographic Protection of SCADA Communications; Part 1: Background, Policies and Test Plan, AGA Report No. 12 (Part 1), Draft 5, Washington, DC (www.gtiservices.org/security/AGA12Draft5r3.pdf), 2005.

[5] American Gas Association, Cryptographic Protection of SCADA Communications; Part 2: Retrofit Link Encryption for Asynchronous Serial Communications, AGA Report No. 12 (Part 2), Draft, Washington, DC (www.gtiservices.org/security/aga-12p2-draft-0512.pdf), 2005.

[6] C. Bowen III, T. Buennemeyer and R. Thomas, Next generation SCADA security: Best practices and client puzzles, *Proceedings of the Sixth Annual IEEE Systems, Man and Cybernetics Information Assurance Workshop*, pp. 426–427, 2005.

[7] T. Brown, Security in SCADA systems: How to handle the growing menace to process automation, *Computing and Control Engineering Journal*, vol. 16(3), pp. 42–47, 2005.

[8] E. Byres and J. Lowe, The myths and facts behind cyber security risks for industrial control systems, presented at the *VDE Congress*, 2004.

[9] W. Clinton, Presidential Decision Directive 63, The White House, Washington, DC (www.fas.org/irp/offdocs/pdd/pdd-63.htm), 1998.

[10] A. Creery and E. Byres, Industrial cyber security for power system and SCADA networks, *Proceedings of the Fifty-Second Annual Petroleum and Chemical Industry Conference*, pp. 303–309, 2005.

[11] J. Fernandez and A. Fernandez, SCADA systems: Vulnerabilities and remediation, *Journal of Computing Sciences in Colleges*, vol. 20(4), pp. 160–168, 2005.

[12] D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn and R. Chandramouli, Proposed NIST standard for role-based access control, *ACM Transactions on Information and System Security*, vol. 4(3), pp. 224–274, 2001.

[13] D. Gaushell and W. Block, SCADA communication techniques and standards, *Computer Applications in Power*, vol. 6(3), pp. 45–50, 1993.

[14] D. Geer, Security of critical control systems sparks concern, *IEEE Computer*, vol. 39(1), pp. 20–23, 2006.

[15] J. Graham and S. Patel, Correctness proofs for SCADA communications protocols, *Proceedings of the Ninth World Multi-Conference on Systemics, Cybernetics and Informatics*, pp. 392–397, 2005.

[16] W. Harrison, N. Hanebutte, P. Oman and J. Alves-Foss, The MILS architecture for a secure global information grid, *CrossTalk: The Journal of Defense Software Engineering*, vol. 18(10), pp. 20–24, 2005.

[17] Instrumentation Systems and Automation Society, Security Technologies for Manufacturing and Control Systems (ANSI/ISA-TR99.00.01-2004), Research Triangle Park, North Carolina, 2004.

[18] Instrumentation Systems and Automation Society, Integrating Electronic Security into the Manufacturing and Control Systems Environment (ANSI/ISA-TR99.00.02-2004), Research Triangle Park, North Carolina, 2004.

[19] T. Kropp, System threats and vulnerabilities (power system protection), *IEEE Power and Energy*, vol. 4(2), pp. 46–50, 2006.

[20] J. Liedtke, On micro-kernel construction, *Proceedings of the Fifteenth ACM Symposium on Operating Systems Principles*, pp. 237–250, 1995.

[21] LynuxWorks (www.lynuxworks.com).

[22] R. McClanahan, SCADA and IP: Is network convergence really here? *IEEE Industry Applications*, vol. 9(2), pp. 29–36, 2003.

[23] A. Miller, Trends in process control systems security, *IEEE Security and Privacy*, vol. 3(5), pp. 57–60, 2005.

[24] M. Naedele and O. Biderbost, Human-assisted intrusion detection for process control systems, *Proceedings of the Second International Conference on Applied Cryptography and Network Security*, 2004.

[25] National Communications System, Supervisory Control and Data Acquisition (SCADA) Systems, Technical Bulletin 04-1, Arlington, Virginia, 2004.

[26] Office of Energy Assurance, 21 Steps to Improve Cyber Security of SCADA Networks, U.S. Department of Energy, Washington, DC, 2002.

[27] P. Oman, E. Schweitzer and D. Frincke, Concerns about intrusions into remotely accessible substation controllers and SCADA systems, *Proceedings of the Twenty-Seventh Annual Western Protective Relay Conference*, 2000.

[28] P. Oman, E. Schweitzer and J. Roberts, Safeguarding IEDs, substations and SCADA systems against electronic intrusions, *Proceedings of the Western Power Delivery Automation Conference*, 2001.

[29] S. Patel, Secure Internet-Based Communication Protocol for SCADA Networks, Ph.D. Dissertation, Department of Computer Engineering and Computer Science, University of Louisville, Louisville, Kentucky, 2006.

[30] President's Commission on Critical Infrastructure Protection, Critical Foundations: Protecting America's Infrastructures, Report Number 040-000-00699-1, United States Government Printing Office, Washington, DC, 1997.

[31] A. Risely, J. Roberts and P. LaDow, Electronic security of real-time protection and SCADA communications, *Proceedings of the Fifth Annual Western Power Delivery Automation Conference*, 2003.

[32] W. Rush and A. Shah, Impact of Information Security Systems on Real-Time Process Control, Final Report, NIST Project SB1341-02-C-081, Gas Technology Institute, Des Plaines, Illinois (www.isd.mel.nist.gov/projects/processcontrol/testbed/GTI_Final_April2005.pdf), 2005.

[33] K. Stouffer, J. Falco and K. Kent, Guide to Supervisory Control and Data Acquisition (SCADA) and Industrial Control Systems Security – Initial Public Draft, National Institute of Standards and Technology, Gaithersburg, Maryland, 2006.

[34] E. Swankoski, N. Vijaykrishnan, M. Kandemir and M. Irwin, A parallel architecture for secure FPGA symmetric encryption, *Proceedings of the Eighteenth International Parallel and Distributed Processing Symposium*, 2004.

[35] A. Wright, Proposal on secure authentication and authorization for remote access to SCADA field equipment, presented at the *Instrumentation Systems and Automation (ISA) Society EXPO*, 2005.

[36] A. Wright, J. Kinast and J. McCarty, Low-latency cryptographic protection for SCADA communications, in *Applied Cryptography and Network Security (LNCS 3089)*, M. Jakobsson, M. Yung and J. Zhou (Eds.), Springer, Berlin-Heidelberg, Germany, pp. 263–277, 2004.