

A Prototype of a Secure Autonomous Bootstrap Mechanism for Control Networks *

Nobuo Okabe Shoichi Sakane
Ubiquitous Lab, Yokogawa Electric Corporation
{*Nobuo.Okabe, Shouichi.Sakane*}@*jp.yokogawa.com*

Masahiro Ishiyama Atsushi Inoue
Corporate R&D Center, Toshiba Corporation
{*masahiro, inoue*}@*isl.rdc.toshiba.co.jp*

Hiroshi Esaki
Graduate School of Information Science and Technology, The University of Tokyo
hiroshi@wide.ad.jp

Abstract

There are many kinds of control networks, which have been used in various non-IP network areas, such as BA (Building Automation), FA (Factory Automation) and PA (Process Automation). They are introducing IP and face the issues of security and configuration complexity. The authors have proposed a model which intends to solve the issues while satisfying restrictions, i.e. small embedded devices, isolated networks and private naming system/name space, which are required when introducing new functionality into existing control networks. Secure bootstrap sequence and device-to-device communication using the chain of trust are the points of the model. This paper shows the practicability of the model through implementing the model experimentally.

1. Introduction

Control networks are different from IP (Internet Protocol) with regard to their history, purposes and technology. There are numerous standards of the control networks, e.g. FOUNDATION fieldbus ¹ [4], PROFIBUS ² [16], MODBUS ³ [13], BACnet ⁴ [1] and LonWorks ⁵ [3], which have been used in various non-IP network areas, such as BA (Building Automation), FA (Factory Automation) and PA (Process Automation). Multiple standards coexist within a single system usually because the system's requirements are diverse. Many standards are introducing IP as a transport technology, e.g. FOUNDATION fieldbus HSE, PROFINet, MODBUS/IP and BACnet/IP.

The following are the issues that the control networks are facing.

- Security

Security of the control networks has not been considered sufficiently. For example, current specifications of FOUNDATION fieldbus and MODBUS do not mention security. BACnet has the capability of network security, however, but it is insufficient [7, 20]. Improvement is on-going [17]. Security of LonWorks supports only server authentication

*This research is supported/funded by the Ministry of Internal Affairs and Communications of Japan.

¹FOUNDATION fieldbus is a registered trademark of the Fieldbus Foundation.

²PROFIBUS is a registered trademark of PROFIBUS International.

³Modbus is a registered trademark of Modicon, Inc.

⁴BACnet is a registered trademark of ASHRAE.

⁵LonWorks is a registered trademark of Echelon Corporation.

using challenge-response before starting a session. The security of LonWorks is weaker than BACnet because mutual authentication and packet based security, i.e. authentication, integrity and confidentiality, are not provided.

They must be as concerned with security as IP networks are because security incidents have occurred on them, and there are concerns for safety of social infrastructure [2, 5, 15].

- Configuration complexity

Devices are manually configured in the fields whereas their user interfaces are not so powerful, like PCs. However, the number of devices are increasing because precision is required in measuring and controlling. For example, a BA system of a large building complex in Japan has 170,000 control points with 16,500 devices⁶. This will present not only the cost of engineering but also the possibility of human errors in the future if a labor-saving mechanism is not introduced.

The following are restrictions that should be accepted when introducing new functionality into the control networks.

- Small embedded devices

The small embedded devices commonly used in the control networks have limited computational performance because of their restricted requirements of cost, physical size and power consumption. Some devices will have more powerful CPUs in the future. At the same time, low-power CPUs will survive because choice of CPU depends upon not only cost performance but also power consumption which has an impact against battery operation or bus width which has an impact on circuit size.

- Isolated network environments

The control networks do not always require connectivities to the Internet even though introducing IP. It is the user's choice whether to connect to the Internet. Therefore, functionalities introduced into them have to work well under an isolated network environment.

- Private naming system and private name space

Information of the control networks, not only the traffic but also device's name, has to be confidential, because the information can help to indicate corporate activities, e.g. the capability of plants. Therefore, the naming system should be closed to the public if operators desire. It is also important not to force device's identity to be global unique if most of the devices should not be accessed by outside. For the above two reasons, DNS is not an appropriate naming system for them.

We have proposed a model [8], which intends to solve the above issues while satisfying the above restrictions when introducing IP into the control networks. It is important for them to inherit existing property when introducing new functionality because they have large property, e.g. specification, operational knowledge and applications. This is the reason the model is a framework whose functions are addressed to either below the application layer or the middle-ware instead of inventing new control network protocols. In this paper, we show the practicability of the model by implementing it experimentally. This paper shows an overview of the model in Section 2, details of the model in Section 3, the prototype system in Section 4, considerations through prototyping in Section 5 and related work in Section 6.

2. Proposed Model

2.1. Network Security

The authors have already studied a security mechanism [14] which can satisfy the restrictions described in Section 1. We will use this mechanism in our proposed model. The following are the features of the security mechanism.

- Communication is protected by IPsec [10] which provides IP packets with confidentiality, integrity and authentication with the other end. IPsec is useful because its enforcement is independent from applications and sharable among them. IPsec is applicable to small embedded devices due to not using public key cryptography.

⁶http://www.echelon.com/about/press/2003/echelon_mori.htm

- It is important for IPsec to share a secret, which is called IPsec SA (Security Association), between both ends. Key exchange protocols will be important in facilitating the sharing of a secret if running IPsec on small embedded devices because these devices do not have a powerful user interface like a PC, which makes manual keying difficult. The security mechanism uses not IKE (the Internet Key Exchange) [6] but KINK (Kerberized Internet Negotiation of Keys) [18] for the key exchange protocol. IKE is the most popular key exchange protocol for IPsec. However, it is not suited to small embedded devices because the Diffie-Hellman key exchange is mandatory. KINK can work well on small embedded devices because KINK is based upon Kerberos⁷ [11], where public key cryptography is not mandated.
- In the security mechanism, a node's identity is in the manner of Kerberos, i.e. a principal-id, which is a combination of a realm name and a principal name.

2.2. Auto-configuration using a Directory Service

To simplify the configuration process, the model provides the device's application layer with an auto-configuration mechanism. The basic ideas of the auto-configuration are 1) to minimize pre-installed information in a device, 2) to acquire most information from servers located in networks. IP address configurations are beyond the scope of the model because it can be done by DHCP (Dynamic Host Configuration Protocol) in IPv4 or RFC2462 in IPv6, with which the model can be combined. The auto-configuration requires not only name/address resolution like DNS but also general data handling, e.g. searching, getting and updating data. We introduce our own directory service named PS (Property Server) [12]. The following are the features of PS.

- It is not a prerequisite condition for PS to connect to the Internet because PS does not require global tree structures like DNS.
- PS maintains a device's attributes as metadata of the device's identity. A typical example is that an IP address IP_{FOO} is an attribute value of an attribute type $ATTR_{IPaddress}$, and the attribute, i.e. the type and the value, belongs to a device's identity FOO as metadata.
- PS supports two types of transactions, i.e. PUT and GET. PUT sets/updates attributes in PS. GET acquires attributes from PS. Any request of transaction has search conditions which designate attributes to be affected. For example, identity/IP address resolution is done by GET transaction, where search conditions is the value of $ATTR_{IPaddress}$ belonging to the name FOO , which returns IP address(es) IP_{FOO} .
- PS's protocol uses XML for the future extension.
- In the proposed model, every node belonging to a system has to use the security mechanism described in Section 2.1. Illegal access to PS by outside can be prohibited with IPsec security policy simply. An access control list can be introduced into PS if accurate restrictions are required.

2.3. Bootstrap Sequence using the Chain of Trust

When considering secure auto-configuration, devices have to discover a trusted PS, then exchange data with PS under secure communication channels. The following bootstrap sequence, which we call the Chain of Trust, satisfy the above requirements.

1. Devices can trust Kerberos server KDC (Key Distribution Center). It is a prerequisite condition of Kerberos.
2. Devices should trust PS which trusted KDC shows.
3. Devices register their own information, e.g. a principal-id and IP address(es), to trusted PS. The information will be used for discovering peers (see Section 2.4).
4. Devices should trust data which trusted PS provides. Then devices can complete the sequence. The communication is protected by IPsec.

Therefore, the minimum information with which a device has to be pre-installed is a principal-id and a key shared with KDC, i.e. a secret key of Kerberos. Other information can be acquired from PS.

⁷Kerberos is a trademark of the Massachusetts Institute of Technology (MIT).

2.4. Device-to-Device Communication

In the control networks, communication is occupied by control messages and notification messages between devices, e.g. controllers, sensors and actuators. Therefore, devices have to discover trusted peers, then to exchange messages with them under secure channels. The Chain of Trust can be applied in that case.

1. Devices search their peers using PS because they have already known trusted PS. (see Section 2.3).
2. Devices should trust peers which trusted PS provides. The communication is protected by IPsec.

3. Details of the Model

Figure 1 shows the bootstrap sequence and the device-to-device communication using the chain of trust. Details of each function is shown in Figure 2 through Figure 6.

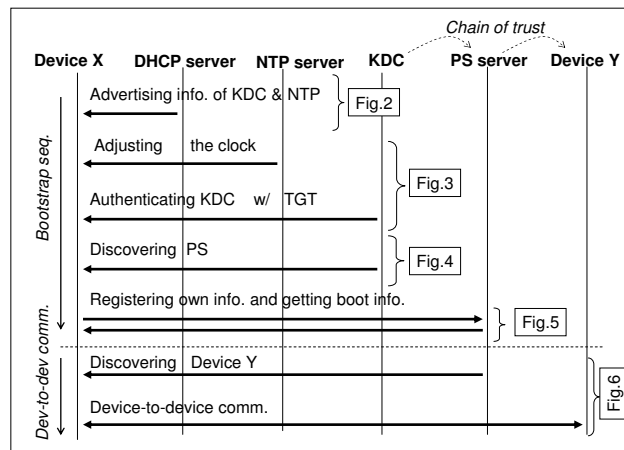


Figure 1. Messages of the proposed model

1. KDC Discovery (KDCD)

DHCP server(s) advertise KDC related information, e.g. KDC’s IP address(es) and realm name(s) where KDC offers authentication services and NTP related information, e.g. NTP server’s IP address(es), (see Figure 2).

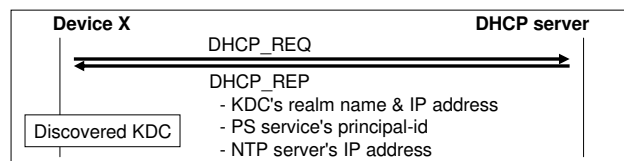


Figure 2. KDC Discovery

2. Authenticating KDC

Device X needs to authenticate KDC advertised by DHCP server (see Figure 3). First, X adjusts its clock with NTP server because Kerberos requires that every device synchronizes its clock to prevent replay attacks. Second, X authenticates KDC, which X learned with DHCP, through verifying a given TGT (Ticket Granting Ticket).

3. PS Discovery (PSD)

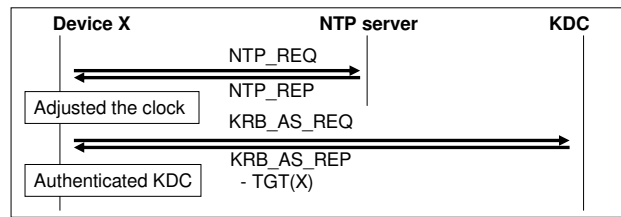


Figure 3. Authenticating KDC

Device X acquires PS’s information from KDC (see Figure 4). X shows its principal-id to KDC. Then KDC returns PS’s information, i.e. PS’s principal-id and IP address(es).

X has to acquire a service ticket for PSD, e.g. TICKET(PSD), prior to the above procedures. KRB_PRIV messages, which need a service ticket, are used for protecting PSD because IPsec/KINK can not be used at this moment.

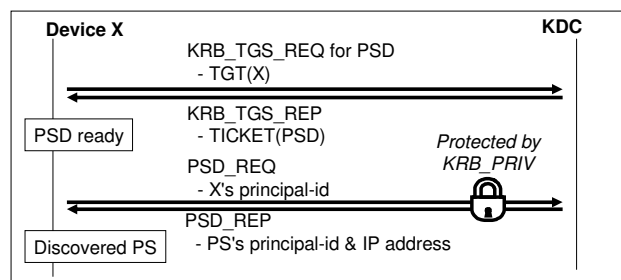


Figure 4. PS Discovery

4. Booting up

Device X registers its own information to PS which is used for device discovery, and acquires its boot data from PS. Then, the bootstrap sequence are completed (see Figure 5).

First, X acquires a service ticket for KINK with PS, e.g. TICKET(KINK w/ PS), from KDC. Second, IPsec is established between X and PS after exchanging KINK messages. Third, X registers its own information, e.g. a principal-id and IP address(es), which other devices or servers can use for discovering. Fourth, X acquires its boot data. Then X completes the bootstrap sequence.

5. Device-to-device communication

Device X can discover device Y using PS, i.e. device discovery, then starts the device-to-device communication with Y (see Figure 6).

First, X discovers Y through PS. A typical example is that X resolves Y’s IP address from Y’s identity like DNS. IPsec between X and PS has already been established at the bootstrap sequence. Second, X acquires a service ticket for KINK with Y, e.g. TICKET(KINK w/ Y), from KDC. Third, IPsec is established between X and Y after exchanging KINK messages. Then the devices can exchange application messages which are protected by IPsec.

4. Prototype System

We implemented the model to examine its practicability, i.e. object code size and performance, experimentally. Table 1 shows the specifications of an experimentally prototyped device, whose CPU is H8/3029 (Renesas Technology Corp.), which has cryptographic hardware in a Xilinx’s FPGA. Renesas’s H8 family is a popular low-end CPU in Japan ⁸. Table 2 shows the specifications of servers which were used for the system.

⁸<http://www.assoc.tron.org/jpn/research/data/survey2003J.pdf>

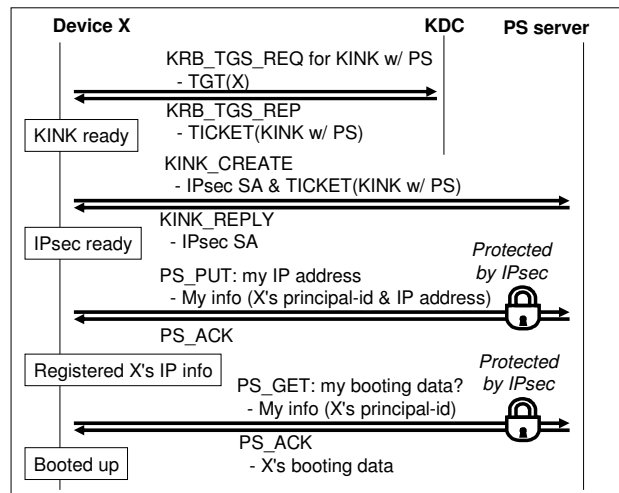


Figure 5. Booting up

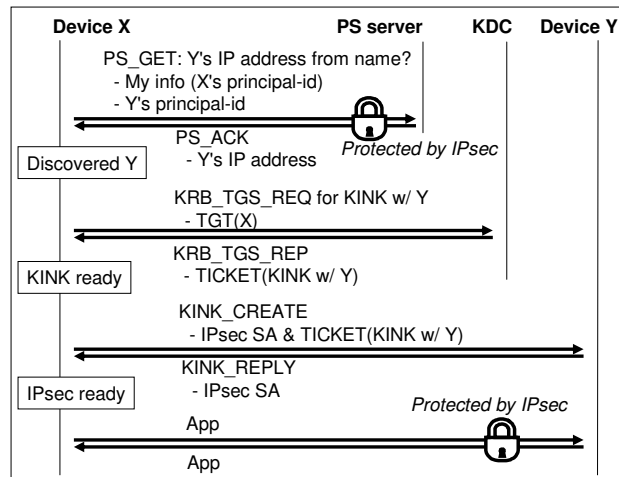


Figure 6. Device-to-device communication

4.1. Object Code Size

Table 3 shows the code size of the device. The total size will be 20K bytes greater if the cryptography, i.e. 3DES and MD5, is implemented by software instead of hardware.

4.2. Performance of the Bootstrap Sequence

Table 4 shows processing time of each function on the device. whose conditions are with and without cryptographic hardware. The values without parentheses mean net processing times, and the values in parentheses mean waiting times from sending a request til receiving a reply. $KINK_I$ and $KINK_R$ mean the processing of KINK initiator and responder. $PS_{PUT,IPsec}$ means PS's PUT transaction of a client side which is for registering device's IP address and includes the overhead of IPsec ESP. $PS_{GET,IPsec}$ means PS's GET transaction of a client side which is for getting device's boot data (512bytes) and includes the overhead of IPsec ESP. The waiting times of $KINK_I$, i.e. the values in parentheses, are varied because they depend upon the performance of a peer. The waiting time 112 msec occurs where a device initiates KINK with PS. The waiting time 213 msec occurs where a device initiates KINK with another device which has cryptographic hardware.

Table 1. The spec. of the device

H/W	H8/3029@20MHz, Crypto H/W@20MHz (3DES,MD5)
OS, IP	uC/OS-II w/ Original IP stack
IPsec	ESP (3DES-CBC,HMAC-MD5)
Kerberos	MIT-1.2.4 based (etype:des-cbc-md5)
KINK	draft-ietf-kink-kink-06 based

Table 2. The spec. of servers

DHCP	CPU:pentium-III@1.2GHz, MEM:128MB, OS:freebsd4.10R
NTP, KDC	CPU:pentium-III@750Mhz, MEM:896MB, OS:linux2.6.8, Kerberos:Heimdal-0.6.2, KINK:racon2
PS	CPU:celeron@1.7GHz, MEM:1GB, OS:linux2.6.8.1

The waiting time 421 msec occurs where a device initiates KINK with another device which does not have cryptographic hardware.

Those values exclude the processing time of IP address configurations, i.e. DHCP in IPv4 or RFC2462 in IPv6, and L2 address resolution, i.e. ARP (Address Resolution Protocol) in IPv4 or ND (Neighbor Discovery) in IPv6.

The bootstrap sequence described in Section 2.3 require the following functions: KDC, NTP, TGT, two TGSs, PSD_{KRB_PRIV} , $KINK_I$, $PS_{PUT,IPsec}$ and $PS_{GET,IPsec}$ (see Figure 2 through Figure 5). Figure 7 shows the processing time of the sequence. The values without parentheses mean net processing times and The values in parentheses mean waiting times.

4.3. Performance of the Device-to-Device Communication

As described in Section 2.4, the device-to-device communication has the overhead (see Figure 6). If the device is an initiator of the communication, the overhead is $PS_{GET,IPsec}$, TGS and $KINK_I$. If the device is a responder of the communication, it is only $KINK_R$. Once IPsec is established between devices, the performance of IPsec is one of major factors.

Figure 8 shows the overhead and IPsec's throughput. The values without parentheses mean net processing times and The values in parentheses mean waiting times. The throughput of IPsec ESP inbound is omitted because both performances are nearly the same.

Table 3. Object code size of the device (K bytes)

Module	Size	Module	Size
OS	50	Kerberos	166
IP (v4/v6)	126	KINK	50
IPsec	8	Crypto	2
		App	16
		total	419

Table 4. Processing time of each function on the device (msec)

crypto H/W	w/	w/o
KDCD	349 (0.5)	349 (0.5)
NTP	27 (0.5)	27 (0.5)
TGT	74 (23)	106 (23)
TGS	195 (24)	294 (24)
PSD _{KRB_PRIV}	239 (2)	373 (2)
KINK _I	263 (112 or 213)	465 (112 or 421)
KINK _R	213 (0)	421 (0)
PS _{PUT,IPsec}	40 (13)	164 (13)
PS _{GET,IPsec}	51 (17)	362 (17)

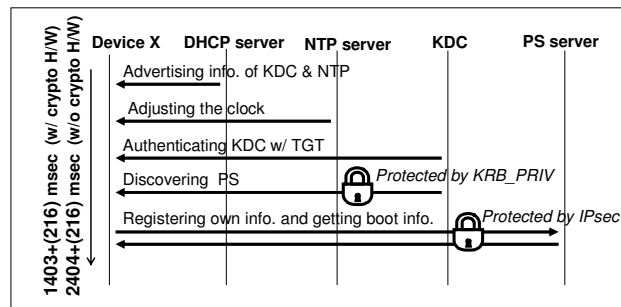


Figure 7. Performance of the Bootstrap Sequence

5. Considerations

5.1. Object Code Size

The size 400K bytes (see Table 3) is not small enough for small embedded devices. For example, there is not enough room for application programs if using internal 512KB FLASH ROM of H8/2029 only. IP stack and Kerberos occupies 30% and 40% of the entire code. Hardwired functions can improve the size. One of the popular examples is iReady's Ethernet MAX, which is the hardwired IP stack. It is a further study item to shrink the object code size.

5.2. Performance of the Bootstrap Sequence

The sequence takes 1619 msec or 2620 msec with or without cryptographic hardware (see Figure 7). It will usually take one or several days to start the entire system if the system is a large one because of starting subsystem by subsystem for making sure. If assuming to start a system described in Section 1 with device-by-device manner, i.e. seventeen thousand devices, within 24 hours, every device will have to start within 5 seconds. The actual margin is longer than the above time because of starting a system in a subsystem-by subsystem manner instead of device-by-device usually. Therefore, the performances of the sequence can be acceptable.

We may have to consider for burst accesses to servers if the system has a large number of devices. For the device side, randomly delayed bootstrap can be a solution. However, the necessity and the validity are future study items. For server side, redundancy can be a solution, i.e. the redundancies of KDCs and PSs. It is not difficult to make KDC redundant [9]. But it is a further study item for PS.

5.3. Performance of Device-to-Device Communication

The overhead of the initiator takes 747 msec or 1566 msec with or without cryptographic hardware (see Figure 8). The overhead happens when both the device starts and the lifetime of Kerberos's tickets or IPsec SAs is expired. The former case

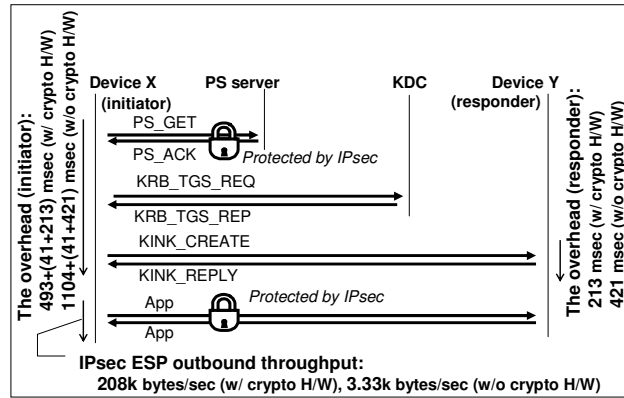


Figure 8. Performance of Device-to-device communication

can be acceptable with the reasons in Section 5.2. The latter case should be considered because the response time of BA or PA system should usually be on the hundred micro-second order. However, the overhead can be acceptable if the lifetimes are long enough, e.g. days, weeks or months, and are tuned operationally. The overhead of the responder can also be acceptable because it is shorter than the initiator's one.

As an example of IPsec's throughput, the processing time for 1024 bytes payload takes 5 msec or 307 msec with or without cryptographic hardware (see Figure 8). Considering the response time, i.e. the hundred micro-second order, the former case is fast enough, but the latter is not. The performance can be improved with AES instead of 3DES if the device cannot introduce cryptographic hardware.

6. Related Work

UPnP⁹ (Universal Plug and Play) [19] is also a framework of device's auto-configuration. UPnP covers not only devices in home networks but also ones in buildings, e.g. HVAC (Heating, Ventilating and Air-Conditioning), temperature sensors or lighting.

UPnP uses SOAP (Simple Object Access Protocol), HTTP and TCP for control messages and GENA (General Event Notification Architecture), HTTP and TCP for notification messages whereas existing control networks usually use UDP for those purposes. It means that they will have to be changed if introducing UPnP. Therefore, the goal of UPnP is different from our proposed model because one of our goals is to minimize the impact on them when introducing an auto-configuration mechanism.

Public key cryptography is mandated for UPnP. So UPnP's applicability to devices is also different from our proposed model.

7. Conclusion

Through implementing the model experimentally, this paper shows the practicability of our proposed model which is intended to solve the issues, i.e. security and configuration complexity, while satisfying the restrictions, i.e. small embedded devices, isolated networks, private name space/naming system and inheriting the property. Secure bootstrap sequence and device-to-device communication using the chain of trust are the points of the model.

The object code size of the prototyped device is not small enough for small embedded devices. It is a further study item to shrink the size. The device's performance of the bootstrap sequence and device-to-device communication can be acceptable if the lifetimes of Kerberos's ticket and IPsec SAs are turned operationally, and if cryptography is implemented reasonably, i.e. cryptographic hardware or faster algorithm than 3DES in software. It is a further study item to make servers redundant.

⁹UPnP is a trademark of the UPnP Implementers Corporation.

References

- [1] ASHRAE. *ANSI/ASHRAE Standard 135-1995, BACnet A Data Communication Protocol for Building Automation and Control Networks*, 1995.
- [2] E. Byres. Plant network security: Can't happen at your site? inTech, ISA, Feb. 2002. http://www.isa.org/Content/ContentGroups/InTech2/Features/20023/February6/Cant_happen_at_your_site_.htm.
- [3] EIA. *EIA/CEA-709.1-B, Control Network Protocol Specification*, 2002.
- [4] Fieldbus Foundation. *FF-581-1.3, FOUNDATION Specification: System Architecture*, 2003.
- [5] J. Gerston. Water and Wastewater Utilities Enhance System Security. In *Texas Water Resources*, volume 27. Texas Water Resources Institute, Dec. 2002.
- [6] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). RFC2409, 1998.
- [7] D. G. Holmberg. BACnet Wide Area Network Security Threat Assessment. NISTIR 7009, NIST, Jul. 2003.
- [8] A. Inoue, M. Ishiyama, et al. A Secured Autonomous Bootstrap Mechanism for Control Networks. Annual Review of Communications, Volume 57, International Engineering Consortium, Nov. 2004.
- [9] C. Kaufman, R. Perlman, and M. Speciner. *Network Security: Private Communication in a Public World*, chapter 10. Prentice Hall, 1995.
- [10] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC2401, 1998.
- [11] J. Kohl and C. Neuman. The Kerberos Network Authentication Service (V5). RFC1510, 1993.
- [12] K. Kubo, J. Murakami, et al. Hybrid Peer-to-Peer System for Network Monitoring of Field Devices. In *SICE Annual Conference 2003 in Fukui*, Aug. 2003.
- [13] MODBUS.ORG. *Modbus Application protocol V1.0*, 2002.
- [14] N. Okabe, S. Sakane, et al. Security Architecture for Control Networks using IPsec and KINK. In *SAINT2005, The 2005 International Symposium on Applications and the Internet*, Jan. 2005.
- [15] K. Poulsen. Slammer worm crashed Ohio nuke plant network. SecurityFocus, Aug. 2003. <http://www.securityfocus.com/news/6767>.
- [16] PROFIBUS International. *IEC 61158, Digital Data Communication for Measurement and Control - Fieldbus for Use in Industrial Control Systems*, 1999.
- [17] D. Robin. IBACnet Security Messages. SPPC 135 WG Document DR-029-6, BACnet committee, Mar. 2004.
- [18] S. Sakane, K. Kamada, et al. Kerberized Internet Negotiation of Keys (KINK). draft-ietf-kink-kink-07.txt, 2005.
- [19] UPnP Forum. *UPnP Device Architecture 1.0, Version 1.0.1*, 2003.
- [20] J. Zachary, R. Brooks, et al. Secure Integration of Building Network into the Global Internet. NIST GCR 02-837, NIST, Oct. 2002.