

Security Architecture for Control Networks using IPsec and KINK *

Nobuo Okabe Shoichi Sakane Kazunori Miyazawa Kenichi Kamada
Ubiquitous Lab, Yokogawa Electric Corporation
{Nobuo.Okabe, Shouichi.Sakane, Kazunori.Miyazawa, Ken-ichi.Kamada}@jp.yokogawa.com

Atsushi Inoue Masahiro Ishiyama
Corporate R&D Center, Toshiba Corporation
{inoue, masahiro}@isl.rdc.toshiba.co.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). These do not incorporate reasonable security mechanisms as they have been mainly used for closed networks. Recently the security of control networks is becoming important because of the popularization of the Internet, the deployment of wireless technologies and the security requirements of such infrastructures. Control networks require security mechanisms which 1) enable end-to-end security that do not depend upon specific network topology, 2) work with multiple control network technologies, and 3) are suited to small embedded devices commonly used in control networks. This paper shows security mechanisms which can meet the above requirements, assuming that IP is applied to the control networks.

1. Introduction

Control networks are different from IP (Internet Protocol) with regard to their history, purposes and technologies. There are numerous technologies of control networks, e.g. FOUNDATION fieldbus¹ [6], PROFIBUS² [24], MODBUS³ [20], BACnet⁴ [1] and LonWorks⁵ [5], which have been used in various non-IP network areas, such as BA (Building Automation), FA (Factory Automation) and PA

*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.

(Process Automation). Multiple technologies coexist within a single system usually because the system's requirements are diverse. Security of control networks have 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, which is insufficient [14, 29]. First, no symmetric key cryptography except DES is supported. Second, the authentication protocol is vulnerable to man-in-the-middle attacks, type flaws, parallel interleaving attacks, replay attacks and implementation dependent flaws. Third, key management is not well defined. Improvement is on-going [25].

Security of LonWorks supports only server authentication using challenge and response before starting a session. The security of LonWorks is weaker than BACnet. First, servers (or responders as the manner of peer-to-peer) can not authenticate clients (or initiators). It is not difficult to impersonate clients. Second, the communication does not provide authentication, integrity and confidentiality of packets. It is far from network security. To be fair, the issues mentioned remain potential before LonWorks introduces IP/Ethernet.

The reasons why those technologies have not supported security sufficiently are the following. a) Original ideas of control networks are to extend their control buses. b) They have been installed in closed networks. c) Control networks have been unfamiliar technologies.

However, a control network must concern itself with security as much as an IP network does for the following reasons. First, they introduce IP as a transport technology, e.g. FOUNDATION fieldbus HSE, PROFINet, MODBUS/IP and BACnet/IP, to be connected with IP networks easily. Second, wireless technologies expose traffic of closed networks to the public space. Third, security incidents have occurred on control networks, and there are con-

cerns for safety of social infrastructures [2, 8, 22].

We believe that security mechanisms for control networks must satisfy the following three requirements.

1. Security is being reconsidered in many control network technologies recently [23, 25]. The Common idea is to rely on the firewall model which assumes specific network topology. However, recent incidents of computer virus show that the firewall model is not always a complete solution. It is challengeable to manage security of normative devices with firewalls. Wireless technologies can expose network traffic behind firewalls easily. Therefore, end-to-end security mechanisms which do not need to assume any specific network topology are necessary.
2. As described above, any control system uses multiple control network technologies, which introduce IP as a transport technology. In another words, control network technologies can be applications. Therefore, security mechanisms for control networks should be implemented below those applications, i.e. in middle-ware or IP layer, to work with every technology
3. The small embedded devices commonly used in control networks have limited computational performance because of their restricted requirements of cost, physical size and power consumption. Some devices will have a more powerful CPU in the future. At the same time, low-power CPUs will survive because choice of CPU depends upon not only cost or performance but also power consumption which has an impact against battery operation or bus width which has an impact against circuit size. Therefore, the security mechanism for control networks should not overload small devices.

This paper shows security mechanisms which meet the above three requirements. This paper evaluates popular cryptographic algorithms in Section 2, cryptographic hardware in Section 3, IPsec performance in Section 4, IPsec's key exchange protocols in Section 5 and shows brief explanation about key exchange protocol KINK in Section 6, proposes a security architecture in Section 7, and shows related work in Section 8.

2. Performance of Cryptography

The computational cost of cryptography must be reasonable for low-end CPUs, i.e. 8-bit or 16-bit CPUs, which are common among control networks. In this section we estimate the computational cost of RSA (the RSA algorithm) which is usually used for authentication, DH (the Diffie-Hellman key exchange) which is usually used for generating session keys, and symmetric key cryptography and hash

functions which are usually used to provide data with privacy and integrity for protecting communications.

First, Figures 1 and 2 show the encrypting/decrypting time of RSA run on a 16-bit CPU, i.e. H8/3048 (Renesas Technology Corp.) which is a popular low-end CPU in Japan.⁶ This program was a part of GnuPG 1.0.7 [10] which was ported to the CPU. Please note that encrypting time is relative to data size, whereas decrypting time is constant because encrypting time does not include the padding specified by PKCS#1 [16] for showing the essential computational cost of RSA, while decryption time must include padding. The figures show that it takes several minutes to decrypt a secret. There is another performance study of the public key cryptography on 8-bit CPU [11] whose results are shown in Table 1. The figures and the table show that RSA can not be suited to low-end CPUs even though there is room for optimization in GnuPG.

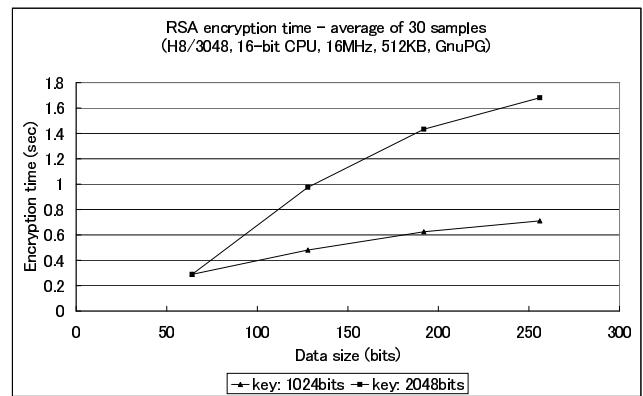


Figure 1. Encrypting performance of RSA

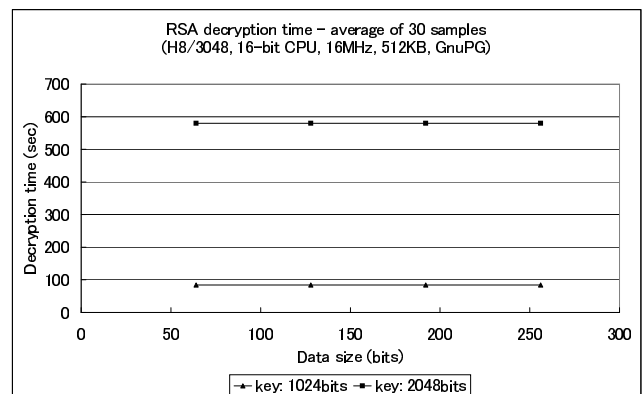


Figure 2. Decrypting performance of RSA

Second, Figure 3 shows the performance of DH run on H8/3048. This estimation uses MODP (modular exponentiation) groups, which is required by IKE (the Internet Key

⁶<http://www.assoc.tron.org/jpn/research/data/survey2003J.pdf>

Table 1. Another estimation of RSA performance [11]

key length (bits)		ATmega128 @ 8MHz (sec.)	CC1010 @ 14.756MHz (sec.)
1024	Encryption	0.43	> 4.48
	Decryption	10.99	106.66
2048	Encryption	1.94	N/A
	Decryption	83.26	N/A

Exchange) [12], rather than elliptic curve groups, which are not mandatory for IKE, because IKE is a candidate for a part of our proposed security system (see Section 5). The processing time in the figure is the sum of the generating time of a DH pair (a secret value and a public value) and the generating time of a DH shared value for each bit length of prime numbers. However, it excludes the generating time of the prime number and the primitive root number. This program is an original code based upon arithmetic libraries in GnuPG 1.0.7 [9]. The figures show that it takes several minutes for DH processing. We can derive a supposition the same as the case of RSA by [11] because the MODP function, which consumes the computational power of the CPU, is common between RSA and DH. Therefore, DH can not be suited to low-end CPUs even though there is room for optimization in our DH.

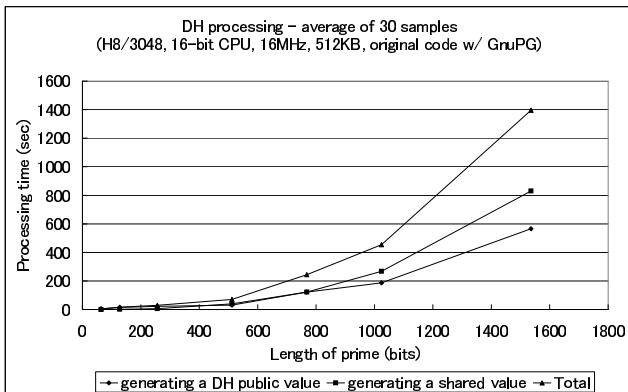


Figure 3. Performance of DH exchange

Third, Figure 4 shows the performance of symmetric key cryptography (AES-128 and 3DES) runs upon a 8051-compatible 8-bit CPU, i.e. DS80C390 (Maxim/Dallas Semiconductor). The 8051 architecture is also a popular low-end CPU. Figure 5 shows the performance of hash functions (MD5 and SHA1). These programs are original

code written in assembler. Figure 4 shows that it takes several seconds to encrypt/decrypt 1000 bytes of data. Figure 5 shows that it takes about one second to generate a hashed value from 1000 bytes of data. There are other performance studies for low-end CPUs, [4] for AES and [7] for MD5 and SHA1. Both studies show that our code of symmetric key cryptography and hash functions have much room for optimization. Therefore, low-end CPUs can achieved reasonable throughput if the right CPU and algorithms are chosen and the algorithms are implemented carefully. Figure 4 and Figure 5 are still useful because of supposing the performance and overhead of IPsec by both figures (see Section 4).

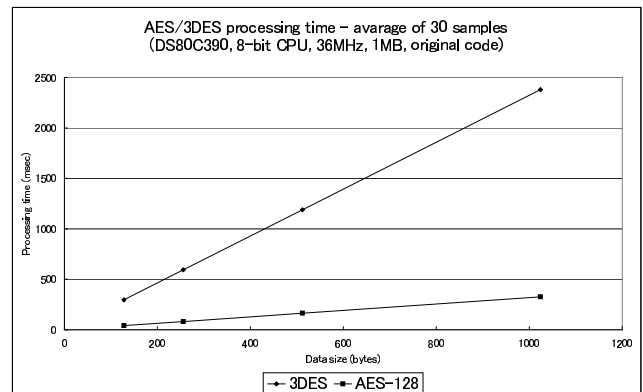


Figure 4. Performance of symmetric key cryptography

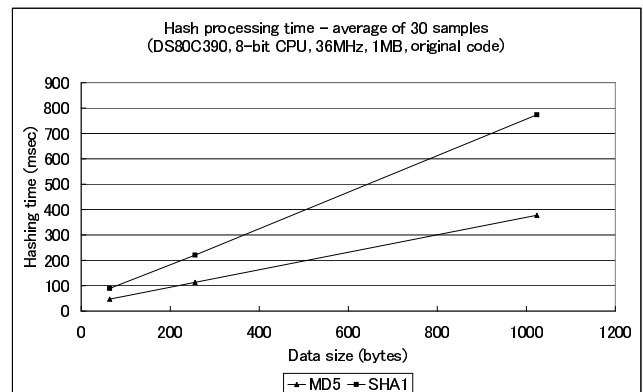


Figure 5. Performance of hash functions

3. Cryptographic Hardware

There are several reasons to introduce cryptographic hardware, e.g. to reduce the code size, to assign more CPU power to applications than to communications or to assure

Table 2. Another estimation of AES performance [4]

Benchmark	DS80C390 equivalent performance (the case of 1024 bytes data)
4065 cycles * 12 clocks/cycle = 48780 clocks	48780 clocks / 36.864MHz * 64 blocks = 84.7 msec

Table 3. Another estimation of hash functions performance [7]

Algorithm	Size (bytes)	Atmega 103 @4MHz (msec)	Atmega 128 @8MHz (msec)
MD5	62-80	10.888	2.722
SHA1	64	31.107	7.777

specific response time. In this section we estimate the gate sizes of cryptographic hardware, i.e. RSA, DH, symmetric key cryptography and hash functions.

Symmetric key cryptography and hash functions need up to ten thousand gates if optimizing for area [21, 26]. RSA and the MODP group of DH use the MODP function $g^x \text{ mod } p$, where g is a primitive root number, p is a prime number and x is a random number. The MODP function for a 1024-bit prime number requires more than one hundred thousands gates [3].

Symmetric key cryptography and hash functions are necessary to maintain the privacy and integrity of packets, even if introducing public key cryptography. It is common among recent security protocols, e.g. SSL, IPsec [18] with IKE. In other words, it is inevitable to implement hardware of symmetric key cryptography and hash functions when introducing hardware of public key cryptography for low-end systems. From an economic point of view, we anticipate that a security system which uses only symmetric key cryptography and hash functions will be more scalable than one using public cryptography, symmetric key cryptography and hash functions because smaller gate size is better for low-end systems.

4. Performance of IPsec

IPsec provides IP packets with privacy and integrity for protecting communications, and is useful because its enforcement is independent from applications. In this section we estimate the performance of IPsec written in assembler, which we implemented on a DS80C390. Figure 6 shows the

outbound performance of IPsec ESP whose cryptographic programs are the same ones used for Figures 4 and 5. Inbound performance is omitted because both performances are nearly the same.

The processing time of IPsec (Figure 6) is almost the sum of the processing time of symmetric key cryptography (Figure 4) and hash functions (Figure 5). This means that the greatest part of the processing time is consumed by cryptography and hash functions. Please note that the performance of symmetric key cryptography and hash functions can be improved as mentioned in Section 2. Therefore, IPsec can achieve reasonable throughput without cryptographic hardware.

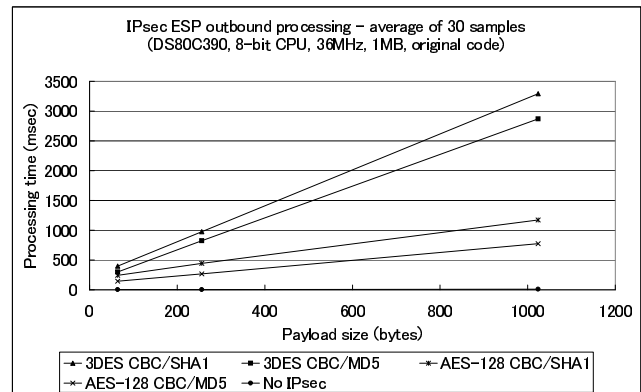


Figure 6. Outbound performance of IPsec ESP

5. Choice of the Key Exchange Protocols

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. IKE, which has been standardized in IETF⁷ IPSEC WG, is the most popular key exchange protocol for IPsec. However, IKE is not suited to small embedded devices as described in Section 2. There is another key exchange protocol for IPsec, i. e. KINK (Kerberosized Internet Negotiation of Keys) [28], which is in a standardizing process of IETF. We expect that KINK can work well on small embedded devices because KINK is based upon Kerberos⁸ [19], which uses symmetric key cryptography and hash functions only. Please see Section 6 for more details about KINK.

⁷IETF (Internet Engineering Task Force): <http://www.ietf.org/>

⁸Kerberos is a trademark of the Massachusetts Institute of Technology (MIT).

6. About KINK

6.1. Features of KINK

IPsec is a security mechanism implemented in the IP layer which offers per-packet basis authentication, privacy and integrity. To enable IPsec, both ends must exchange IPsec SA which includes IP addresses of the both ends, algorithms, shared secret and the lifetime of SA. KINK is a key exchange protocol to exchange IPsec SA. KINK has the following features:

- KINK, which is based upon Kerberos, is a key exchange protocol to share IPsec SA between both ends.
- Kerberos gives tickets which include authentication mechanism and session keys, to entities. Both ends exchanging KINK are authenticated with a Kerberos ticket whose communications are protected by the given session key.
- A Kerberos server, i.e. KDC (Key Distribution Center), and an entity which uses Kerberos service must share a secret key. KDC manages identities and shared secret keys of all devices centrally.
- An identity of Kerberos consists of a principal name and a realm name instead of an IP address. A principal name specifies a name of a device and a realm name specifies a domain of Kerberos. KINK uses a DNS-style name, i.e. FQDN (Fully Qualified Domain Name), as a principal name. Therefore, the identity of KINK is *FQDN@realm name*, where @ is a delimiter. KINK does not presume any naming system. It is an implementation matter to choose naming systems, e.g. DNS, some directory system, etc.
- KINK does not mandate public key cryptography since it is based upon Kerberos.

6.2. KINK Exchange

This section shows an example of KINK's behavior which is simplified for explanation. Table 4 shows the players of the example, where devices X and Y belonging to realm MY establish IPsec with KINK, a DNS server DNS-FOO maintains A/AAAA-records of X and Y, and a Kerberos server KDC works for realm MY. Figure 7 shows the network of the example.

It takes three steps to establish IPsec SA between X and Y using KINK. In the following, IPsec SA will be called SA, communication between X and Y is protected by IPsec.

Table 4. Example players

Player	Description	Player	Description
X	Device belonging to realm MY IP address: IPx Kerberos identity: X.FOO.ORG@MY Secret key: Kx	KDC	Kerberos server for realm MY IP address: IPkdc Managing secret keys: X.FOO.ORG@MY: Kx Y.FOO.ORG@MY: Ky
Y	Device belonging to MY realm IP address: IPy Kerberos identity: Y.FOO.ORG@MY Secret key: Ky	DNS-FOO	DNS server for zone FOO.ORG Managing A/AAAA-records: X.FOO.ORG: IPx Y.FOO.ORG: IPy

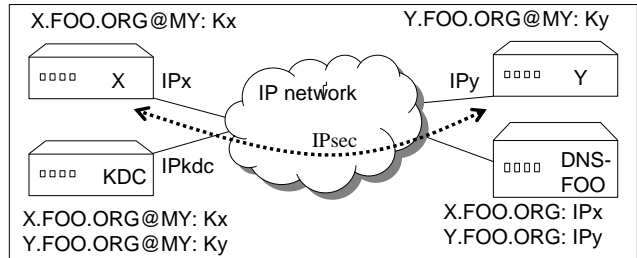


Figure 7. Example network

1. TGT exchange: TGT (Ticket-Granting Ticket) exchange is a part of Kerberos exchange. A TGT is required for any Kerberos service within a specific domain, i.e. a realm. (see Figure 8).
2. TGS exchange: TGS (Ticket-Granting Service) exchange is a part of Kerberos exchange. A TGS is required for a peer to establish IPsec using KINK exchange. (see Figure 9).
3. KINK exchange: X and Y establish SA whose parameters are derived from TGS using KINK exchange. (see Figure 10).

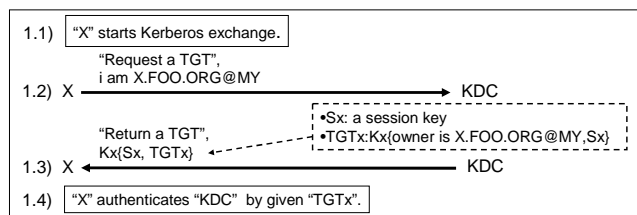


Figure 8. Kerberos exchange of TGT

7. Proposed System

7.1. System Architecture

Figure 11 is our proposal for a security architecture which is suitable for control networks. Device-to-device

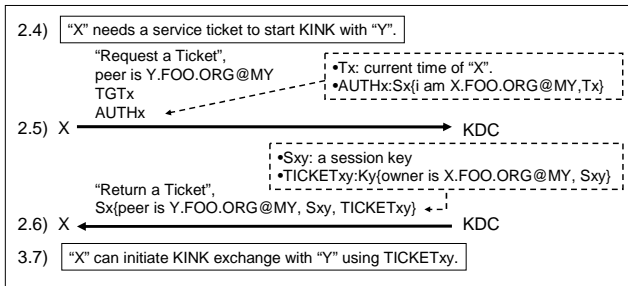


Figure 9. Kerberos exchange of TGS

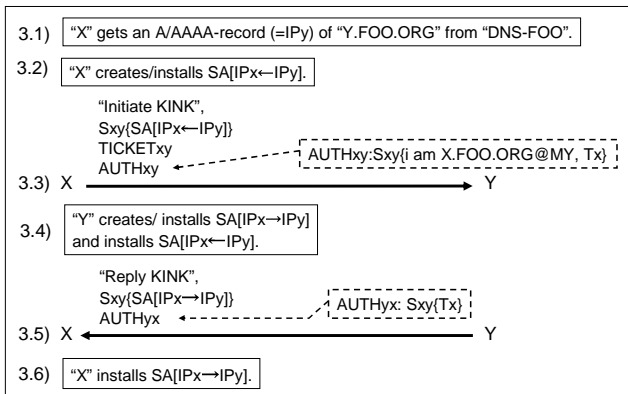


Figure 10. KINK exchange

communication on IP is protected by IPsec whose key exchange protocol is KINK. The Kerberos server gives KINK-specific tickets to the devices for mutual authentication. Each device in the system has symmetric key cryptography and hash functions which can be optionally implemented by hardware if the device's computational capabilities are inadequate. This system requires only symmetric key cryptography and hash functions, and does not need public key cryptography, e.g. RSA or DH, which are known for their high computational consumption.

7.2. Pros and Cons of the Proposed System

The proposed system has the following advantages:

- The cost of cryptography (computational cost and gate size of hardware) of this system is smaller than one which introduces public key cryptography. This means that this system can be applied on lower end devices.
- All shared secrets are stored in Kerberos server(s), which makes the entire system manageable.

The system has the following disadvantages:

- The system must introduce Kerberos server(s) which can increase management cost of control systems, especially security management. The major premise of

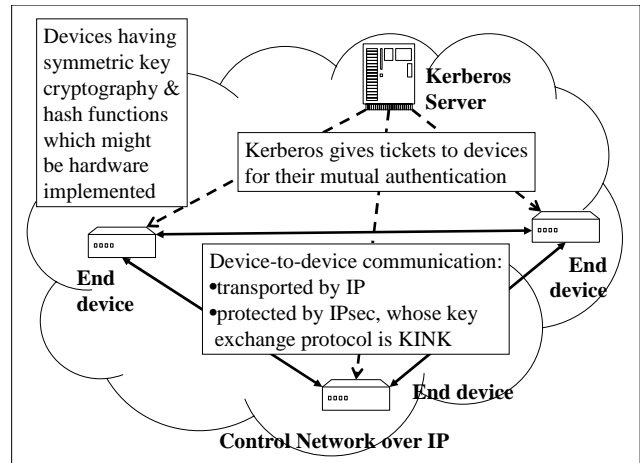


Figure 11. Proposed System

the proposed system is that every kerberos server and device is tamper-resistant. If a secret key in a device is stolen, it will be easy to impersonate the device. If a database in a Kerberos server is stolen, it will be easy to impersonate any device.

- A Kerberos server can be a single point of failure. However, it is not difficult to make Kerberos servers redundant because Kerberos's database is updated infrequently [17] and several implementation supports distributed database, e.g. LDAP (Lightweight Directory Access Protocol) [13].
- Another concern is interoperability with servers or PCs. When they can speak IP, IPsec and IKE are commonly used, but KINK is not at this moment. However, it is not difficult to implement KINK for them because KINK is a simple protocol whose specification has been opened to the public by IETF.

From a practical point of view, we deem the disadvantages to be acceptable or surmountable. We are currently developing a prototype system to estimate its validity.

8. Related Work

Related works concerning cryptographic performance on embedded CPUs are shown in Section 2. This section shows other related work which deal with the usability of embedded devices from a system security viewpoint.

- It is essential that Kerberos share a secret key between Kerberos server and a device securely. How should the key be installed in the device? There are two ways to install the key. One is to ship a device with the key of factory default. The other is to create/install the key by the customer/user.

In the former case, the factory default key has caused many security issues, e.g. well-known secret or leakage. In the latter case, the key must not be changed by an unauthorized person although the capability of devices are limited, e.g. limited CPU power, user interface or communication channel. The *resurrecting duckling* security policy model [27] gives us hints for the issue.

- Our proposed architecture presumes that every device knows appropriate KDC and naming system. But how can they know those? Manual installation will not work if there are a huge number of devices. The authors are studying a secure bootstrap mechanism [15] which is based upon the proposed architecture.

9. Summary

We have shown a security architecture which is suited to control networks by satisfying the following requirements (see Section 1 for more details):

1. The architecture enables end-to-end security.
2. The architecture can be implemented as application independent.
3. The architecture is suited to low-end CPUs or small embedded devices.

One of the important points of this study is the performance evaluation of real systems because competition between IPsec and KINK for cryptographic hardware can stall the system's throughput. That will be done on the prototype system which we are currently developing.

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] chipSign A/S. CS-9010 IP core. <http://www.chipsign.com/downloads/PBF-9010-01-02.pdf>.
- [4] J. Daemen and V. Rijmen. The Block Cipher Rijndael. In *Third International Conference, CARDIS'98*, volume LNCS 1829, pages 277–284, 1998.
- [5] EIA. *EIA/CEA-709.1-B, Control Network Protocol Specification*, 2002.
- [6] Fieldbus Foundation. *FF-581-1.3, FOUNDATION Specification: System Architecture*, 2003.
- [7] P. Ganesan, R. Venugopalan, et al. Analyzing and modeling encryption overhead for sensor network nodes. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 151–159, 2003.
- [8] J. Gerston. Water and Wastewater Utilities Enhance System Security. In *Texas Water Resources*, volume 27. Texas Water Resources Institute, Dec. 2002.
- [9] GnuMP. A free library for arbitrary precision arithmetic. <http://www.swox.com/gmp/>.
- [10] GnuPG. A RFC2440 (OpenPGP) compliant application. <http://www.gnupg.org/>.
- [11] N. Gura, A. Patel, et al. Comparing Elliptic Curve Cryptography and RSA on 8-Bit CPUs. In *Cryptographic Hardware and Embedded System - CHES 2004*, LNCS 3156, pages 119 – 132, 2004.
- [12] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). RFC2409, 1998.
- [13] J. Hodges and R. Morgan. Lightweight Directory Access Protocol (v3): Technical Specification. RFC3377, 2002.
- [14] D. G. Holmberg. BACnet Wide Area Network Security Threat Assessment. NISTIR 7009, NIST, Jul. 2003.
- [15] 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.
- [16] B. Kaliski. PKCS #1: RSA Encryption Version 1.5. RFC2313, 1998.
- [17] C. Kaufman, R. Perlman, and M. Speciner. *Network Security: Private Communication in a Public World*, chapter 10. Prentice Hall, 1995.
- [18] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC2401, 1998.
- [19] J. Kohl and C. Neuman. The Kerberos Network Authentication Service (V5). RFC1510, 1993.
- [20] MODBUS.ORG. *Modbus Application protocol V1.0*, 2002.
- [21] Ocean Logic Pty Ltd. The summary of encryption cores. <http://www.ocean-logic.com/des.htm>.
- [22] K. Poulsen. Slammer worm crashed Ohio nuke plant network. SecurityFocus, Aug. 2003. <http://www.securityfocus.com/news/6767>.
- [23] Process Control Security Requirements Forum. NIST. <http://www.isd.mel.nist.gov/projects/processcontrol/>.
- [24] PROFIBUS International. *IEC 61158, Digital Data Communication for Measurement and Control - Fieldbus for Use in Industrial Control Systems*, 1999.
- [25] D. Robin. IBACnet Security Messages. SPPC 135 WG Document DR-029-6, BACnet committee, Mar. 2004.
- [26] H. Shimizu, F. Sano, et al. Implementation of SPN block cipher. Technical Report of IEICE ISEC 2001-55, IEICE, Sep. 2001.
- [27] F. Stajano and R. Anderson. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In *Security Protocols, 7th International Workshop Proceedings*, LNCS, pages 172–194, 1999.
- [28] M. Thomas and J. Vilhuber. Kerberized Internet Negotiation of Keys (KINK). draft-ietf-kink-kink-06.txt, 2003.
- [29] J. Zachary, R. Brooks, et al. Secure Integration of Building Network into the Global Internet. NIST GCR 02-837, NIST, Oct. 2002.