JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 29, 281-298 (2013)

Hybrid Key Management Architecture for Robust SCADA Systems*

DONGHYUN CHOI[1], HANJAE JEONG[1], DONGHO WON[2] AND SEUNGJOO KIM[3,+]
[1]*Samsung Electronics*
Suwon-si, Gyeonggi-do, 443-742 Korea
[2]*Information Security Group*
Sungkyunkwan University
Suwon-si, Gyeonggi-do, 440-746 Korea
[3]*Center for Information Security Technologies*
Korea University
Seoul, 136-701 Korea

Recently, as a demand for connecting Supervisory Control and Data Acquisition (SCADA) systems to open networks increases, the study of SCADA system security becomes an issue. Many researchers have proposed key management schemes for SCADA systems. However, previous studies lack the proper considerations for availability. In this paper, we build up cryptographic security requirements for robust SCADA systems. In addition, we propose a hybrid key management architecture for robust SCADA systems which supports replace protocol for availability and reduces the number of keys to be stored in a master terminal unit.

Keywords: SCADA systems, power system security, key management, cryptography, protocol

1. INTRODUCTION

Modern industrial facilities such as electric power generating plants have command and control systems. These industrial command and control systems are commonly called as Supervisory Control and Data Acquisition (SCADA) systems.

As a demand for connecting SCADA systems to open networks increases, the study of SCADA system security becomes an issue. Many researchers have proposed key management schemes for SCADA. However, previous studies lack the proper considerations for availability. Namely, they do not have a solution for when the main device has broken down. In addition, since many SCADA system devices are remote from the control center, they are physically insecure. Therefore, the devices need to periodically update the security keys which they store. However the computation and communication costs of this update process increase as both the number of vulnerable devices and keys increases, so SCADA systems need to reduce the number of keys transmitted for security and efficiency.

In this paper, we propose hybrid key management architecture for a robust SCADA

Received January 22, 2011; revised August 22, 2011; accepted November 11, 2011.
Communicated by Chin-Laung Lei.

[+] Corresponding author: Seungjoo Kim.

[*] This research was supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract UD100002KD and by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program (NIPA-2012-H0301-12-3007) supervised by the NIPA (National IT Industry Promotion Agency).

system which supports the replace protocol for availability and reduces the number of keys to be stored in a master terminal unit (MTU). This is because the proposed scheme applied the public key cryptosystem between MTU and sub-MTU which have high performance and the symmetric key cryptosystem between sub-MTU and remote terminal unit (RTU) which has low performance.

2. REQUIREMENTS

2.1 Cryptographic Security Requirement for SCADA

In this section, we rebuild the cryptographic security requirements based on standards and reports.

(a) Access Control

The SCADA system uniquely identifies and authenticates organizational users and devices [1].

(b) Availability

The availability of a SCADA system is more important than confidentiality, because an unavailable SCADA system can cause physical damage or threaten human life [2]. Usually, SCADA systems have spare devices, because SCADA systems should be designed to be always on. If the main device is broken down, then it should be replaced with a spare device as soon as possible.

(c) Confidentiality

The data transmitted between nodes should be protected by encryption.

(d) Cryptographic Key Establishment and Management

When cryptography is required and employed within the control system, the organization establishes and manages cryptographic keys using automated mechanisms with supporting procedures or manual procedures [1]. The procedures require Broadcasting/ Multicasting [3], Backward Secrecy (BS) [4], Group Key Secrecy (GKS) [5], Forward Secrecy (FS) [4], Key Freshness, and Perfect Forward Secrecy (PFS) [6].

(e) Integrity

It is critical that messages between nodes are not tampered with, and that no new message is inserted [2] since message modification and injection can cause physical damage. Therefore, the SCADA system should ensure the integrity of the transmitted message.

(f) Public Key Infrastructure

The organization issues public key certificates under an appropriate certificate policy or obtains public key certificates under an appropriate certificate policy from an approved service provider.

(g) Number of Keys

Since many SCADA system devices are remote from the control center, they are physically insecure. Therefore, the devices need to periodically update the security keys which they store. In addition, if a device has many keys and the device is compromised, then other devices which have those keys become vulnerable too. Therefore, each device which has keys must perform the update process. Since the computation and communication costs of this update process increase as both the number of vulnerable devices and keys increases, so SCADA systems need to reduce the number of keys stored on each device for security and efficiency.

2.2 SCADA Performance Requirements

A SCADA system needs to interact with devices in real time. In Bowen *et al*. [7], SCADA transactions must have a time delay of no more than 0.540 seconds. In Boulay and Reilly [8], the time latency should be less than 0.900 seconds for states and alarms. So, our proposed architecture for SCADA communication must match the shortest time delay requirement of no more than 0.540 seconds.

Generally, a SCADA communications link operates at low speeds such as 300 to 19,200 baud rate [9]. In the Modbus implementation guide, default baud rate is now 19200, and if that cannot be implemented then the default baud rate is 9600 [10]. Therefore, we assume a requirement of a 9600 baud rate in this paper.

2.3 SCADA Network Topology Requirements

When the SCADA system was first developed, the system architecture was based on a mainframe. Remote devices communicated directly with the MTU by serial data transmission. The second generation SCADA systems took advantage of developments and improvement in systems miniaturization and local area networking (LAN) technology to distribute the processing load across multiple systems. Thus, when a local MTU or human machine interface (HMI) had trouble, the devices could be promptly replaced. The current SCADA system is close to that of the second generation [11]. In this paper, we assume that a SCADA system's topology is second generation.

3. PREVIOUS SCHEMES ANALYSIS

3.1 Previous Schemes

In this section, we estimate and analyze the costs of the ten previous schemes. Total time delay is the sum of the group key setup time, message encryption/decryption time, certificate verification time and data transmission time. Table 1 shows group key setup time, message encryption/decryption time, certificate verification time and data transmission time, and Table 2 shows the total time delay. In Table 2, highlighted boxes show that the time delay is less than 0.540 seconds. Table 3 shows the security comparison between previous schemes. In the table, "O" means the scheme guarantee the requirement, "×" means the scheme does not guarantee the requirement, "Δ" means the scheme guarantee the requirement but it is inefficient, and "-" means not applicable to the scheme.

Table 1. Time delay.

	Key setup time (sec)	Message encryption/decryption time (sec)	Certificate verification time (sec)	Data communication time (sec) by baud rate									
				115200 (baud)	38400 (baud)	19200 (baud)	9600 (baud)	4800 (baud)	2400 (baud)	1200 (baud)	600 (baud)	300 (baud)	110 (baud)
ASKMA+	0.000021	0.000017	0	0.020000	0.060000	0.120000	0.240000	0.480000	0.960000	1.920000	3.840000	7.680000	20.94545
ASKMA	0.000021	0.000017	0	0.020000	0.060000	0.120000	0.240000	0.480000	0.960000	1.920000	3.840000	7.680000	20.94545
RSA	0.005574	0.000017	0.00014	0.037778	0.113333	0.226667	0.453333	0.906667	1.813333	3.626667	7.253333	14.50666	39.56363
BD	0.007994	0.000017	0.00448	0.660000	1.980000	3.960000	7.920000	15.84000	31.68000	63.36000	126.7200	253.4400	691.2000
TGDH	0.008554	0.000017	0.00098	0.197778	0.593333	1.186667	2.373333	4.746667	9.493333	18.98666	37.97333	75.94666	207.1272
GDH	0.106104	0.000017	0.00448	1.193333	3.580000	7.160000	14.32000	28.64000	57.28000	114.5600	229.1200	458.2400	1249.745
CKD	0.055584	0.000017	0.00014	0.037778	0.113333	0.226666	0.453333	0.906666	1.813333	3.626666	7.253333	14.50666	39.56363
AGKA WMN	0.009164	0.000017	0.00448	0.056944	0.170833	0.341666	0.683333	1.366666	2.733333	5.466666	10.93333	21.86666	59.63636
TT	0.008224	0.000017	0.00238	0.046667	0.140000	0.280000	0.560000	1.120000	2.240000	4.480000	8.960000	17.92000	48.87272
NCKW	0.002030	0.000017	0.00035	0.064444	0.193333	0.386666	0.773333	1.546666	3.093333	6.186666	12.37333	24.74666	67.49090

Signature Algorithm: RSA 1024 Signature, Certificate Form at: X509 v3,
The number of MT: 33, Size of Diffie-Hellman parameter p: 1024 bit, Size of Diffie-Hellman parameter q: 1024
See more details about the analysis environment in section.

Table 2. Total time delay.

	Total time delay (sec) by baud rate									
	115200 (baud)	38400 (baud)	19200 (baud)	9600 (baud)	4800 (baud)	2400 (baud)	1200 (baud)	600 (baud)	300 (baud)	110 (baud)
ASKMA+	0.020039	0.060039	0.120039	0.240039	0.480039	0.960039	1.920039	3.840039	7.680039	20.94549
ASKMA	0.020039	0.060039	0.120039	0.240039	0.480039	0.960039	1.920039	3.840039	7.680039	20.94549
RSA	0.039331	0.114887	0.228220	0.454887	0.908220	1.814887	3.628220	7.254887	14.50822	39.56519
BD	0.666454	1.986454	3.966454	7.926454	15.84645	31.68645	63.36645	126.7265	253.4465	691.2065
TGDH	0.204791	0.600347	1.193680	2.380347	4.753680	9.500347	18.99368	37.98035	75.95368	207.1343
GDH	1.299455	3.686121	7.266121	14.42612	28.74612	57.38612	114.6661	229.2261	458.3461	1249.852
CKD	0.037949	0.113505	0.226838	0.453505	0.906838	1.813505	3.626838	7.253505	14.50684	39.56381
AGKA WMN	0.065966	0.179855	0.350688	0.692355	1.375688	2.742355	5.475688	10.94235	21.87569	59.64538
TT	0.053348	0.146681	0.286681	0.566681	1.126681	2.246681	4.486681	8.966681	17.92668	48.87941
NCKW	0.066331	0.195220	0.388553	0.775220	1.548553	3.095220	6.188553	12.37522	24.74855	67.49280

Signature Algorithm: RSA 1024 Signature, Certificate Form at: X509 v3,
The number of MT: 33, Size of Diffie-Hellman parameter p: 1024 bit, Size of Diffie-Hellman parameter q: 1024
See more details about the analysis environment in section.

(a) ASKMA

In ASKMA, Choi *et al.* proposed a key management scheme suitable for secure SCADA communication using a logical key hierarchy [12]. The overall performance of ASKMA has many advantages compared to previous studies, but it may be less efficient during the multicast communication process. Furthermore, ASKMA lacks the proper availability considerations.

(b) ASKMA+

Choi *et al.* proposed the ASKMA+ protocol which is more efficient and secure compared to previous schemes [13]. ASKMA+ reduces the number of keys stored and provides efficient multicast and broadcast communication. However, as shown in Table 3, ASKMA+ does not satisfy the availability requirement.

Table 3. Security requirements.

	ASKMA+	ASKMA	RSA	BD	TGDH	GDH	AGKA WMN	CKD	TT	NCKW
Broadcasting	○	○	○	○	○	○	○	○	○	○
Multicasting	△	△	○	○	○	○	○	○	○	○
Group Key Secrecy	○	○	○	○	○	○	○	○	○	○
Forward Secrecy	○	○	○	○	○	○	○	○	○	○
Backward Secrecy	○	○	○	○	○	○	○	○	○	○
Perfect Forward Secrecy	–	–	×	○	○	○	○	○	○	○
Key Freshness	○	○	○	○	○	○	○	○	○	○
Availability	×	×	×	×	×	×	×	×	×	×

(c) GDH

The Group Diffie-Hellman (GDH) [14] protocol is a contributory group key agreement protocol which generalizes upon the well-known 2 party Diffie-Hellman key exchange. However, since the GDH protocol has a lot of exponentiation and heavy traffic, this protocol is not suitable for a SCADA system. As shown in Table 2, GDH cannot support a 115200 baud rate. Furthermore, GDH does not satisfy SCADA network topology requirements since GDH needs to communicate between each RTU.

(d) RSA

The Rivest, Shamir and Adleman (RSA) [15] protocol is a public key cryptosystem. The basic idea is that the group controller encrypts a group key with each member's RSA public key and sends it to each member. As shown in Table 2, RSA can support a 9600 baud rate. However, RSA does not guarantee perfect forward secrecy and lacks the proper considerations for availability.

(e) CKD

The Centralized Key Distribution (CKD) [16] protocol is a simple group key management scheme. The group key is always generated by the group controller. Following each membership change, the controller generates a new secret key and distributes it securely to the group. As shown in Table 2, CKD can support a 9600 baud rate. However, CKD lacks the proper considerations for availability.

(f) TGDH

The Tree-based Group Diffie-Hellman (TGDH) [17] protocol is an adaptation of key trees in the context of a fully distributed, contributory group key agreement. TGDH computes a group key derived from individual contributions of group members using a logical binary key tree. As shown in Table 2, TGDH cannot support a 9600 baud rate. In addition, TGDH lacks the proper considerations for availability and does not satisfy the SCADA network topology requirement since TGDH needs to communicate between each device.

(g) BD

Burmester and Desmedt [18] presented a practical interactive conference key distribution system. The main idea in BD is to distribute the computation among members. In a BD protocol, since each member sends some values to all other members, communication traffic is heavy. Therefore, BD is not suitable for a low-speed SCADA system. As shown in Table 2, BD cannot support an 115200 baud rate. In addition, BD lacks the proper considerations for availability and does not satisfy the SCADA network topology requirement since it needs each device to communicate with every other device.

(h) TT

Tan and Teo [19] proposed a group key agreement protocol based on the Schnorr signature and the BD scheme. To provide efficiency, they combine the computational efficiency of the Schnorr scheme and the round efficiency of the BD scheme. As shown in Table 2, TT can support a 9600 baud rate. However, TT lacks the proper considerations for availability. In addition, TT does not satisfy the SCADA network topology requirement since the protocol needs to communicate between each device.

(i) NCKW

Nam *et al.* proposed group key agreement protocol based on factoring [20]. Their protocol needs a constant round communication to generate a group key with optimal message complexity. As shown in Table 2, NCKW cannot support a 9600 baud rate. In addition, NCKW lacks the proper considerations for availability.

(j) AGKA WMN

The Authenticated Group Key Agreement protocol for Wireless Mesh Networks (AGKA WMN) [21] generates a session key based on Diffie-Hellman key exchange over an insecure channel and is designed to reduce computation and communication costs. As shown in Table 2, AGKA WMN cannot support a 9600 baud rate. In addition, AGKA WMN lacks the proper considerations for availability.

4. THE PROPSOED KEY MANAGEMENT PROTOCOL

In the previous section we analyzed 10 schemes. We found that ASKMA, ASKMA+, RSA and CKD satisfied the performance requirements, but all of these schemes lacked proper considerations for availability. Namely, if the main device breaks down, then previous protocols cannot solve this problem. In addition, RSA does not guarantee perfect forward secrecy.

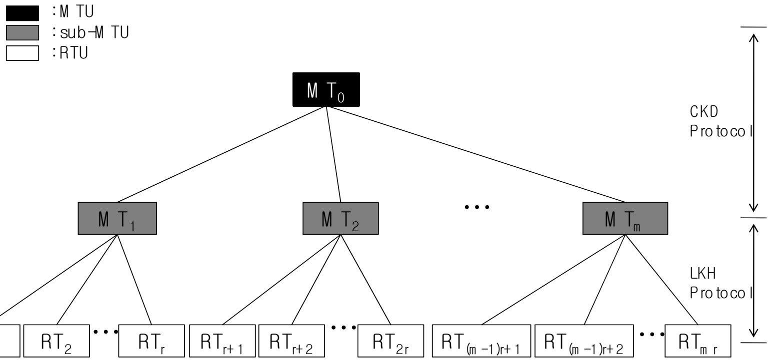
In this section, we propose hybrid key management architecture for robust SCADA systems. In a SCADA system, MTUs and sub-MTUs have reasonable computational resources as desktop computers. Therefore, we apply a public key cryptosystem between an MTU and a sub-MTU.

Since the proposed scheme applied a public key cryptosystem between MTU and sub-MTU which have high performance and the symmetric key cryptosystem between SUB-MTU and RTU which has low performance, the proposed scheme reduces the number of keys stored in each MTU. Furthermore, the proposed scheme includes a replace protocol. A replace protocol operates when the main device has broken down and the SCADA system has switched to a reserve device allowing continuous work.

4.1 Notations

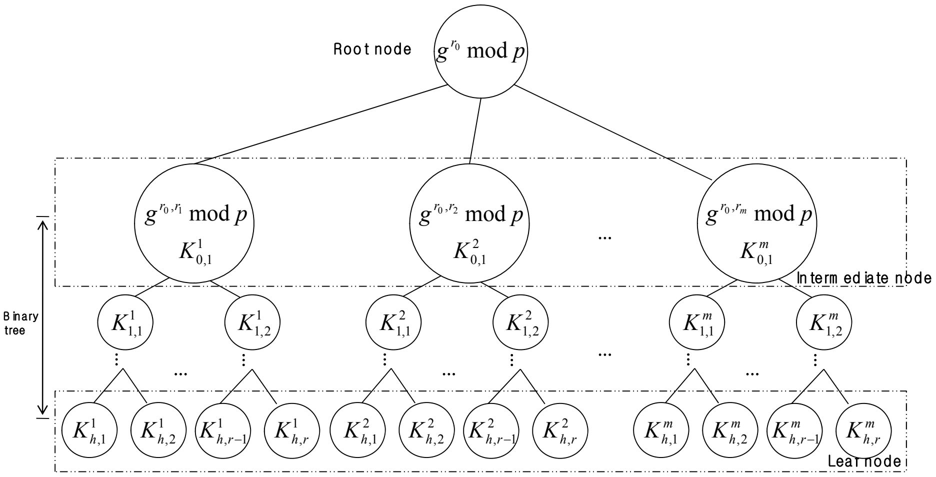
The following notation is used throughout this paper.

- m: number of sub-MTUs;
- r: maximum number of RTUs per sub-MTU;
- GM: nonempty set of nodes. This set is divided into two disjoint subsets MT and RT, *i.e.*, $GM = MT \cup RT$;
- RT: $RT = \{RT_1, \ldots, RT_{m \cdot r}\}$ is the set of RTU;
- MT: $MT = \{MT_0, \ldots, MT_m\}$ is the nonempty set of MTU or sub-MTU;
- g: generator of the subgroup of order q;
- p: prime number such that $p = kq + 1$ for some small $k \in N$;
- q: order of the algebraic group;
- r_i: MT_i's random number $r_i \in Z_q$;
- IK_i: MT_i's intermediate key;
- $K^k_{i,j}$: MT_k's jth key at level i in a binary tree.
- $E_{g^{r_i r_j}}(K_g)$: $(K_g)^{g^{r_i r_j}} \bmod p$.



CKD and LKH protocols are connected through the Iolus framework

Fig. 1. System architecture.



RT_i: knows keys from leaf node to intermediate node
$MT_i (i \neq 0)$: knows all keys which are on the path from the leaf node to root node.

Fig. 2. Key hierarchy.

4.2 Initialization

Toward the goal, we implement the CKD protocol, Iolus framework and Logical key structure as shown Fig. 1. A proposed protocol has two parts MTs and RTs. MTs make a group key by the CKD protocol, and RTs are constructed as a logical key hierarchy structure. Each RT_i knows keys from leaf node to intermediate node as shown in Fig. 2. Each MT_i ($i \neq 0$) knows all keys which are on the path from the leaf node to the root node as shown in Fig. 2. The MT and RT are connected through the Iolus framework. The MT_0 (MTU) plays the role of a Group Security Controller. Therefore, the MT_0 manages the entire group and the group key between the MT_0 and MT_i ($1 \leq i \leq m$). The MT_i ($1 \leq i \leq m$) plays the role of a Group Security Intermediary. It manages the subgroup key of its subgroup consisting of r RTs. The architecture of RT and connection of RT and MT are same as in the ASKMA+ protocol.

The group key K_g is always generated by MTU. Initialization of the protocol runs as follows:

- Step 1: MT_0 (MTU) selects random r_0, computes $g^{r_0} \bmod p$ and broadcasts it to the group with a digital signature.
- Step 2: Each member MT_i ($i \in [1, m]$), checks the validity of the digital signature, selects random r_i, computes $g^{r_i} \bmod p$ and sends it to the MTU with a digital signature.
- Step 3: Each member MT_i ($i \in [1, m]$) and MT_0 compute $g^{r_0 r_i} \bmod p$.
- Step 4: MT_0 checks the validity of the digital signatures, generates a group key K_g which is a random value, computes $IK_i = (K_g)^{g^{r_0 r_i}} \bmod p$ ($i \in [1, m]$), and signs it.

In the protocol, the devices can previously compute until step 4. When the group member is fixed, the protocol runs as follows:

- Step 5: MT_0 sends IK_i back to MT_i ($i \in [1, m]$) with a digital signature.
- Step 6: Upon receipt of the message, each member MT_i ($i \in [1, m]$) computes $K_g = K_g^{g^{r_0 r_i}/g^{r_0 r_i}} \bmod p$.

4.3 Join

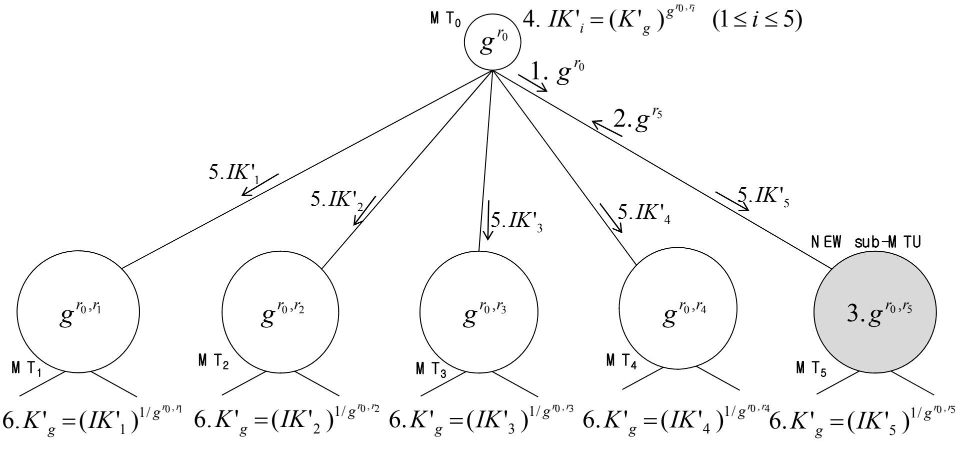
In this subsection, we present the join protocol. If a new sub-MTU device MT_{m+1} join the SCADA system, then the protocol runs as follows:

- Step 1: MT_0 sends $g^{r_0} \bmod p$, which was generated in the initialization phase, to a new device MT_{m+1} with a digital signature.
- Step 2: The new device MT_{m+1} checks the validity of the digital signature, selects random r_{m+1}, computes $g^{r_{m+1}} \bmod p$ and sends it to the MT_0 with a digital signature.
- Step 3: The new device MT_{m+1} and MT_0 compute $g^{r_0 r_{m+1}} \bmod p$.
- Step 4: MT_0 checks the validity of the digital signatures, generates a new group key K'_g computes $K'_i = K'_g{}^{g^{r_0 r_i}/ g^{r_0 r_i}} \bmod p$ ($i \in [1, m + 1]$), and signs it.
- Step 5: MT_0 sends IK'_i ($i \in [1, m + 1]$) back to MT_i with a digital signature.
- Step 6: Upon receipt of the message, each member MT_i($i \in [1, m + 1]$) computes $K'_g = K'_g{}^{g^{r_0 r_i}/g^{r_0 r_i}} \bmod p$.

In principle, r_i should be updated all the time, but we can improve efficiency by repeatedly using r_i like SSL's "session cache mode" [22]. Although our protocol reuses r_is, each group member cannot know the other group member's $g^{r_0 r_i}$, since our protocol uses exponentials to compute IK'. It can be applied to a leave and replace protocol as well as a join protocol.

Fig. 3 shows a simple illustrative example of the join protocol, where a new sub-MTU is MT_5 and $m = 4$.

The RTU join protocol performs the same procedure as the ASKMA+ protocol.



We omit the "mod p".

Fig. 3. Simple illustrative example of join protocol.

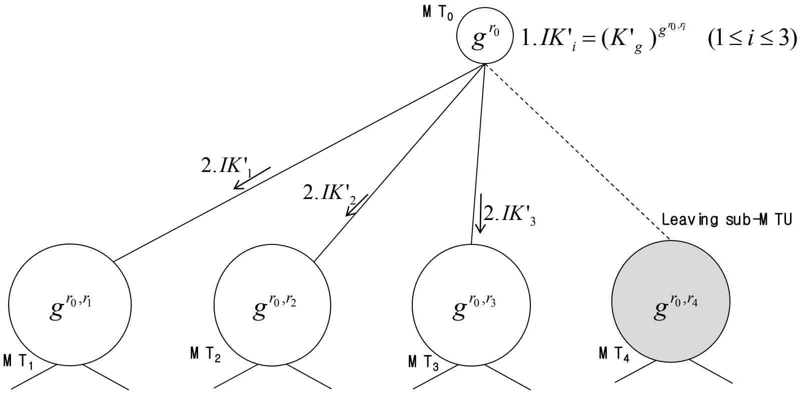
4.4 Leave Protocol

In this subsection, we represent the leave protocol. If a SUBMTU device MT_j leaves the SCADA system, the protocol runs as follows:

- Step 1: MT_0 generates a new group key K'_g, computes $IK'_i = (K'_g)^{g^{r_0 r_i}} \bmod p$ ($i \in [1, m]$ and $i \neq j$), and signs it.
- Step 2: MT_0 sends IK'_i ($i \in [1, m]$ and $i \neq j$) to MT_i with a digital signature.
- Step 3: Upon receipt of the message, each member MT_i ($i \in [1, m]$ and $i \neq j$) computes $K'_g = K'_g{}^{g^{r_0 r_i}/g^{r_0 r_i}} \bmod p$.

Fig. 4 shows a simple illustrative example of the leave protocol, where a leaving sub-MTU is MT_4 and $m = 4$.

The RTU leave protocol performs the same procedure as the ASKMA+ protocol.



$3. K'_g = (IK'_1)^{1/g^{r_0, r_1}}$ $3. K'_g = (IK'_2)^{1/g^{r_0, r_2}}$ $3. K'_g = (IK'_3)^{1/g^{r_0, r_3}}$

We omit the "mod p".

Fig. 4. Simple illustrative example of leave protocol.

4.5 Replace Protocol

In this subsection, we present the replace protocol for supporting availability. If some devices in a SCADA system fail, then these devices should be replaced with the reserve devices. In this case, leave processes and join processes are performed at the same time. Thus the replace protocol is a combination of leave and join protocols.

If the sub-MTU device MT_n fails, then MT_n should be switched to the reserve sub-MTU device. First of all, the SCADA system runs a leave protocol, so MT_0 generates a new group key K'_g, encrypts it with each device's key $g^{r_0 r_i} \bmod p$ ($i \in [1, m]$ and $i \neq n$) and then sends it. In the second place, the SCADA systems runs a join protocol, so MT_0 and the new MT'_n make a new key $g^{r_0 r'_n} \bmod p$ and share the new group key K'_g. The replace protocol runs as follows:

- Step 1: MT_0 generates a new group key K'_g, computes $IK'_i = (K'_g)^{g^{r_0 r_i}}$ ($i \in [1, m]$ and $i \neq$

n), and signs it.

- Step 2: MT_0 sends IK'_i ($i \in [1, m]$ and $i \neq n$) to MT_i with a digital signature.
- Step 3: Upon receipt of the message, each member MT_i ($i \in [1, m]$ and $i \neq n$) computes $K'_g = {K'_g}^{g^{r_0 r_i}/g^{r_0 r_i}} \bmod p$.
- Step 4: MT_0 sends $g^{r_0} \bmod p$ to the reserve sub-MTU MT'_n with a digital signature.
- Step 5: MT'_n checks the validity of the digital signature, selects a new random r'_n, computes $g^{r'_n} \bmod p$ and sends it to the MT_0 with a digital signature.
- Step 6: MT'_n and MT_0 compute $g^{r_0 r_n} \bmod p$.
- Step 7: MT_0 checks the validity of the digital signatures, generates a new group key K'_g, computes $IK'_n = (K'_g)^{g^{r_0 r_i}} \bmod p$, and signs it.
- Step 8: MT_0 sends IK'_n to MT'_n with a digital signature.
- Step 9: Upon receipt of the message, MT'_n computes $K'_g = {K'_g}^{g^{r_0 r_n}/g^{r_0 r_n}} \bmod p$.

Fig. 5 shows a simple illustrative example of the leave protocol, where a broken device is $MT4$ and $m = 4$.

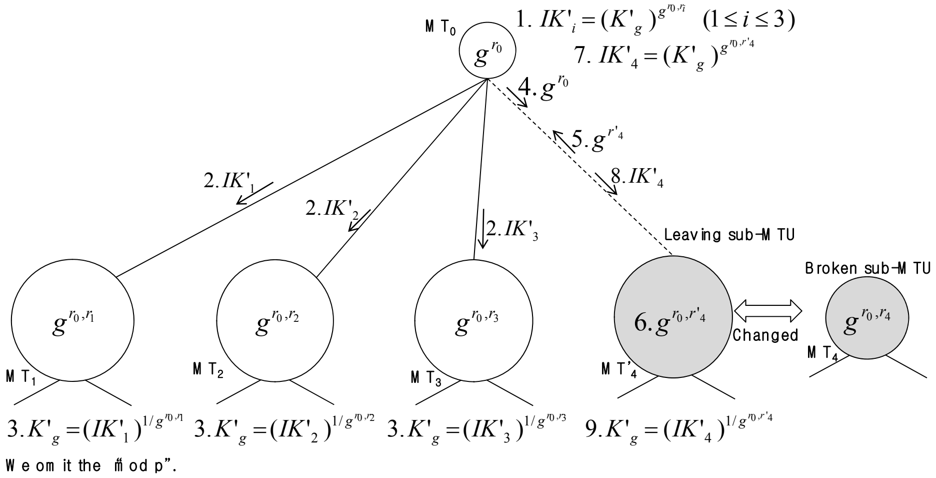


Fig. 5. Simple illustrative example of replace protocol.

4.6 Data Encryption

In this subsection, we present the data encryption algorithms for unicast, broadcast, and multicast. For the freshness of the session key, we use a time variant parameter (TVP). The TVP is a combination of the timestamp and the sequence number. In unicast, the session key for data encryption is generated the following equation:

$$SK_u = H(K^k_{hj}, TVP). \tag{1}$$

K^k_h is a leaf node's key where h is a height of the tree. The data is encrypted with session key SK_U.

In broadcast and multicast, the session key for data encryption should be generated using shared information by every member. The generation of the session key for broadcast and multicast uses the following equation:

$$SK_B = H(K_g, TVP). \tag{2}$$

The key K_g is a shared key among all group members or some members of the group.

4.7 Key Freshness

In this subsection, we present the period to update the keys in the RTUs. Since the RTUs in general are located remotely, they are physically insecure. Therefore, they need to periodically update the keys which they store. However, if the time interval between updating the keys is too short, then it causes more time delay in SCADA communication. Thus, we should find an appropriate period to update the keys which satisfies communication efficiency and security. So, we define the *QoS* function to find the period [23].

$$QoS = CI + SI \tag{3}$$

CI and *SI* stand for communication index and security index. *CI* is computed based on the time delay caused by to update the keys in the RTUs. Assume that T is the period of communication in the SCADA system and δ is the time delay caused by updating keys, and *CI* is computed below:

$$CI = (T - \delta)/T. \tag{4}$$

Since the period to update the keys is in inverse proportion to the δ, we can modify the above formula:

$$CI = (T - \delta)/T = (T - (k/t_p))/T. \tag{5}$$

where k is a constant and t_p is the time between updating the current and next keys.

SI is calculated by the probability of a successful attack to the RTUs. Since a successful attack to the RTUs is recognized as an independent event in real life, we can employ a Poisson process to express the event [23]:

$$\frac{(\lambda t)^n}{n!} e^{-\lambda t_p}, \; n = 0, 1, \ldots \tag{6}$$

where n is the number of the events during the time (= t), and is the mean of the number of the successful attacks to the RTUs. Our security goal is that the successful attack to the key in the RTUs should not occur between updating the current and next keys. So we can derive the formula below with $n = 0$ and $t = t_p$.

$$SI = e^{-\lambda t_p} \tag{7}$$

In [23], λ represents the mean of the number of every possible attack to the SCADA network. However, we can restrict the target of attacks to the keys in the RTUs. Then, we can separate the reason for attacks into either a logical error of the scheme to update the keys in the RTUs or an error of implementation. Some examples of attacks caused by logical errors are forward secrecy, backward secrecy and so on. Attacks caused by an error of implementation can be separated into invasive attacks on RTUs and non-invasive attacks on RTUs. An example of an invasive attack on RTUs is reverse engineering of the hardware module of the RTUs. An example of a non-invasive attack on the RTUs is a side channel attack, or reverse engineering of the software in the RTUs.

We can re-calculate *SI*

$$SI = e^{-(\lambda_l + \lambda_i + \lambda_{ni})t_p} \tag{8}$$

where λ_l is the mean of the number of successful attacks caused by logical errors, λ_i is the mean of the number of successful invasive attacks, and λ_{ni} is the mean of the number of successful non-invasive attacks caused by an error in implementation. However, our scheme has any logical error according to the security analysis in section 5.2. So, we can assign λ_l of our scheme to 0.

Finally, the *QoS* function can be expressed by t_p.

$$QoS = \frac{T - k/t_p}{T} + e^{-(\lambda_l + \lambda_i + \lambda_{ni})t_p} \tag{9}$$

To maximize the *QoS* function, a differentiation of the *QoS* function at a t_p should be 0.

$$\frac{dQoS(t_p)}{dt_p} = \frac{k}{T.t_p^{\,2}} - (\lambda_l + \lambda_i + \lambda_{ni})e^{-(\lambda_l + \lambda_i + \lambda_{ni})t_p} = 0 \tag{10}$$

Thus, we can find the optimal period for updating the key in the RTUs.

5. ANALYSIS

5.1 Performance Analysis

In this section, we estimate and analyze the cost of the proposed scheme. We assume the analysis environment as follows:

- The number of *MT*: 33
- Size of Diffie-Hellman parameter *p*: 1024 bit
- Size of Diffie-Hellman parameter *q*: 160 bit
- Run time of exponentiation: 0.00008s
- Run time of RSA-1024 signing: 0.00148s
- Run time RSA-1024 verification: 0.00007s
- Run time AES-128/CBC: 0.000009s
- Signature algorithm: RSA 1024 Signature
- Certificate format: X.509 v3

The case of the number of *MT*s, we referred to Bowen *et al.* [7]. We choose Diffie-Hellman parameters *p* and *q* by recommendation of Barker *et al.* [24]. For run time, we make reference to Crypto++ 5.6.0 Benchmarks [25]. We choose RSA and X.509 v3, since RSA and X.509 v3 are the most commonly used public key cryptosystem scheme and certificate format.

In general, the message size of a SCADA system is less than 1000 bit [7]. Therefore, message encryption/decryption time is 0.000018s. Commonly, symmetric key size is 128 bit, so key encryption and decryption time is 0.0000034s. Group key setup time is 0.00015s because group key setup phase has 1 exponentiation operation and 1 verification operation. Therefore, the sum of these values and transmission time is total time delay. Table 4 shows the total time delay for the proposed scheme. The proposed scheme satisfied the performance requirements because the total delay time is 0.333505 sec with 9600 baud rate.

Table 4. Total time delay of the proposed scheme.

	Total time delay (sec) by baud rate									
	115200 (baud)	38400 (baud)	19200 (baud)	9600 (baud)	4800 (baud)	2400 (baud)	1200 (baud)	600 (baud)	300 (baud)	110 (baud)
Proposed Scheme	0.037949	0.113505	0.226838	0.453505	0.906838	1.813505	3.626838	7.253505	14.50684	39.56381

Table 5. Number of keys to be stored in a device.

	SKE	SKMA	ASKMA	ASKMA+	Proposed Scheme
MTU	m(1+r)	m(1+r)	2m-1+mr	2m-1	m+2
Each SUB-MTU	1+r	1+r	$r+1+\log_2 m$	$2r+\log_2 m$	2r+1
Each RTU	1	1	$2+loge_2m$	$1+\log_2 r$	$1+\log_2 r$

m is the number of SUB-MTUs, r is the maximum number of RTUs per SUB-MTU

In the proposed scheme, the number of keys stored in an MTU is less than that in the other schemes. In Table 5, we compare the number of keys stored in an MTU for SKE, SKMA, ASKMA, ASKMA+ and the proposed scheme.

Fig. 6 compares the total computational time based on the number of multicast target nodes with 5kb messages ($r = 128$ and $m = 4$).

5.2 Security Analysis

In this section we show the security analysis for the proposed scheme. In our hybrid key management architecture, we apply CKD between an MTU and a SUB-MTU, and LKH between SUB-MTU and RTU. Therefore if CKD and LKH scheme are secure, our scheme is secure.

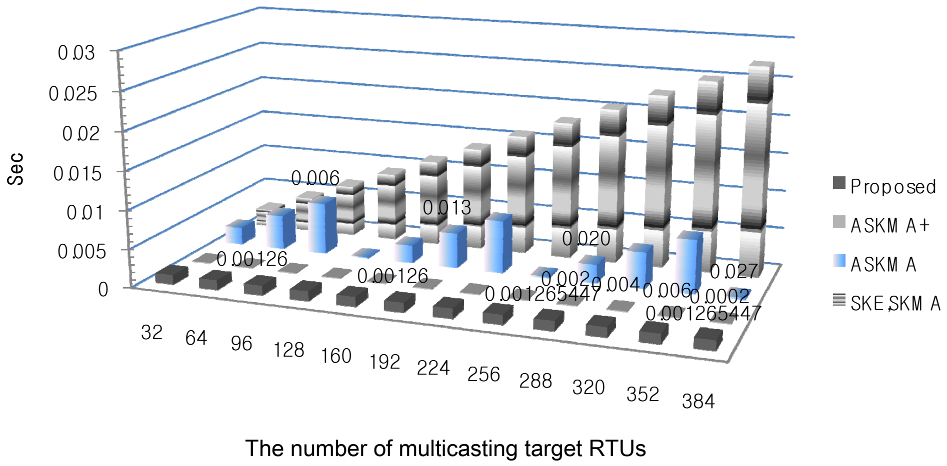


Fig. 6. Comparison of the total computational time based on the number of multicast target nodes with 5kb message ($r = 238$ and $m = 4$).

Theorem 1 Assuming the Decision Diffie-Hellman assumption and Discrete Logarithm are satisfied, CKD provides key independence, key confirmation, perfect forward secrecy and resistance to known key attacks.

Proof: CKD always generates the group key K_g by one member (in our scheme the one member is MT_0) and distributes it securely to the group. As shown in [16, 26], CKD relies on the Decision Diffie-Hellman assumption and Discrete Logarithm Problem and provides the same level of security as GDH based on [27], as far as key independence, key confirmation, perfect forward secrecy and resistance to known key attacks. Therefore CKD is secure.

Theorem 2 If all keys assigned to the node of the key-tree are distinct, LKH scheme is secure.

Proof: According to [28], if the new keys assigned to new leaves and the keys assigned to the nodes of the key-tree during a revoke/join operation are distinct among them, from all the others, and from previously used and deleted ones (*i.e.*, all keys assigned to the node are distinct), then LKH scheme is secure.

Therefore, by combining Theorems 1 and 2 our scheme is secure.

6. CONCLUSION

In this paper, we propose hybrid key management architecture for a robust SCADA system which supports replace protocol and reduces the number of keys to be stored in a MTU, because the proposed scheme applies the public key cryptosystem between MTU and SUB-MTU which have high performance and the symmetric key cryptosystem between SUB-MTU and RTU which has low performance.

REFERENCES

1. Homeland Security, “Catalog of control systems security: Recommendations for standards developers,” September 2009.
2. R. D. Colin, C. Boyd, J. Manuel, and G. Nieto, “KMA − A key management architecture for SCADA systems,” in *Proceedings of the 4th Australasian Information Security Workshop*, Vol. 54, 2006, pp. 138-192.
3. American Gas Association, “Cryptographic protection of SCADA communications part 1: Background, policies and test plan,” Technical Report 12-1 Draft 5 revision 3, 2005, http://www.gtiservice.org/security/.
4. T. Hardjono, B. Cain, and B. Doraswamy, “A framework for group key management for multicast security,” *IETF Internet Draft*, 2001.
5. A. Perrig, D. Song, and J. D. Tygar, “ELK, a new protocol for efficient large group key distribution,” in *Proceedings of IEEE Symposium on Security and Privacy*, 2001, pp. 247-262.
6. W. Diffie, P. C. van Oorschot, and M. J. Wiener, “Authentication and authenticated key exchanges,” *Designs*, *Codes and Cryptography*, Vol. 2, 1992, pp. 107-125.
7. C. L. Bowen, T. K. Buennemeyer, and R. W. Thomas, “Next generation SCADA security: Best practices and client puzzles,” in *Proceedings of IEEE Workshop on Information Assurance and Security United States Military Academy*, 2005, pp. 426-427.
8. M. Boulay and R. O’Reilly, “Implementing real-time statistical analysis of power quality in a distributed generating environment,” Cooper Industries, poster.
9. A. K. Wright, J. A. Kinast, and J. McCarty, “Low-latency cryptographic protection for SCADA communications,” in *Proceedings of the 2nd International Conference on Applied Cryptography and Network Security*, 2004, pp. 263-277.
10. Modbus, “MODBUS over serial line specification and implementation guide V1.02,”
11. Communication Technologies, “Supervisory control and data acquisition (SCADA) systems,” *NCS Technical Information Bulletin* 04-1, 2004.
12. D. Choi, H. Kim, D. Won, and S. Kim, “Advanced key management architecture for secure SCADA communications,” *IEEE Transactions on Power Delivery*, Vol. 24, 2009, pp. 1154-1163.
13. D. Choi, S. Lee, D. Won, and S. Kim, “Efficient secure group communications for SCADA,” *IEEE Transactions on Power Delivery*, Vol. 25, 2010, pp. 714-722.
14. G. Ateniese, M. Steiner, and G. Tsudik, “Authenticated group key agreement and friends,” in *Proceedings of the 5th ACM Conference on Computer and Communications Security*, 1998, pp. 17-26.
15. R. L. Rivest, A. Shamir, and L. Adleman, “A method of obtaining digital signature and public key cryptosystem,” *Communications of the ACM*, Vol. 21, 1978, pp. 120-126.
16. Y. Amir, Y. Kim, C. Nita-Rotaru, J. Schultz, J. Stanton, and G. Tsudik, “Secure group communication using robust contributory key agreement,” *IEEE Transactions on Parallel and Distributed Systems*, Vol. 15, 2004, pp. 468-480.
17. Y. Kim, A. Perrig, and G. Tsudik, “Tree-based group key agreement,” *ACM Transactions on Information and System Security*, Vol. 7, 2004, pp. 60-96.
18. M. Burmester and T. Desmedt, “A secure and efficient conference key distribution system,” in *Proceedings of EUROCRYPT ’94*, LNCS, Vol. 950, 1995, pp. 275-286.

19. C. H. Tan and J. C. M. Teo, "An authenticated group key agreement for wireless networks," in *Proceedings of IEEE Wireless Communications and Networking Conference*, Vol. 4, 2005, pp. 2100-2105.
20. J. Nam, S. Cho, S. Kim, and D. Won, "Simple and efficient group key agreement based on factoring," in *Proceedings of International Conference Computational Science and Its Applications*, LNCS, Vol. 3043, 2004, pp. 645-654.
21. Z. A. Jin, "A provable secure authenticated group key agreement protocol for WMNs," Ph.D. Thesis, Department of Computer Science and Engineering, Kyungpook National University, 2008.
22. A. O. Freier, P. Karlton, and P. C. Kocher, "The SSL protocol version 3.0," *Internet Draft*, November 18, 1996.
23. D. J. Kang, J. J. Lee, B. H. Kim, and D. Hur, "Proposal strategies of key management for data encryption in SCADA network of electric power systems," *International Journal of Electrical Power and Energy Systems*, Vol. 33, 2011, pp. 1521-1526.
24. E. Barker, D. Johnson, and M. Smid, "Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography," *NIST Special Publication*, 800-56A, 2007.
25. "crypto++ 5.6 benchmarks," http://www.cryptopp.com/benchmarks.html.
26. Cliques, http://sprout.ics.uci.edu/past_projects/cliques/downloa d.html.
27. M. Steiner, G. Tsudik, and M. Waidner, "Key agreement in dynamic peer groups," *IEEE Transactions on Parallel and Distributed Systems*, 2000, pp. 769-780.
28. P. D'Arco and A. de Santis, "Optimizing SD and LSD in presence of non-uniform probabilities of revocation," in *Proceedings of International Conference Information Theoretic Security*, LNCS, Vol. 4883, 2009, pp. 46-64.

Donghyun Choi received his B.E. degree in Electrical and Computer Engineering from Sungkyunkwan University, Korea, in 2005 and an M.S. degree in Computer Science from Sungkyunkwan University, Korea, in 2007 and a Ph.D. degree in Mobile Systems Engineering at Sungkyunkwan University, Korea in 2010. He is currently engineer of the Samsung electronics. His current research interests are in the areas of cryptography, SCADA, mobile security, and DRM.

Hanjae Jeong received his B.E. degree in Electrical and Computer Engineering from Sungkyunkwan University, Korea, in 2006 and an M.S. degree in Computer Science from Sungkyunkwan University, Korea, in 2008 and a Ph.D. degree in Mobile Systems Engineering at Sungkyunkwan University, Korea in 2012. He is currently engineer of the Samsung electronics. His current research interests are in the areas of cryptography, SCADA, and mobile security.

Dongho Won received his B.E., M.E., and Ph.D. degrees from Sungkyunkwan University in 1976, 1978, and 1988, respectively. After working at the Electronics & Telecommunications Research Institute (ETRI) from 1978 to 1980, he joined Sungkyunkwan University in 1982, where he is currently a Professor of the School of Information and Communication Engineering. His interests are cryptology and information security. He was the President of the Korea Institute of Information Security & Cryptology (KIISC) in 2002.

Seungjoo Kim received his B.S. (1994), M.S. (1996), and Ph.D. (1999) in information engineering from Sungkyunkwan University (SKKU) in Korea. Prior to joining the faculty at Korea University (KU) in 2011, He served as Associate Professor of School of Information and Communication Engineering at SKKU for 7 years. Before that, He served as Director of the Cryptographic Technology Team and the (CC-based) IT Security Evaluation Team of the Korea Information Security Agency (KISA) for 5 years. Now he is Full Professor of Graduate School of Information Security at KU, and a member of KU's Center for Information Security Technologies (CIST). Also, he has served as an executive committee member of Korean E-Government, and advisory committee members of several public and private organizations such as National Intelligence Service of Korea, Digital Investigation Advisory Committee of Supreme Prosecutors' Office, Ministry of Justice, The Bank of Korea, ETRI (Electronic and Telecommunication Research Institute), and KISA (Korea Information Security Agency), *etc*. His research interests include cryptography, information security and information assurance. He is a corresponding author.