Hybrid Key Management Architecture for Robust SCADA Systems*

DONGHYUN CHOI¹, HANJAE JEONG², DONGHO WON³ AND SEUNGIOO KIM¹

¹Samsung Electronics
Suwon-si, Gyeonggi-do, Korea
²Information Security Group
Sungkyunkwan University
Suwon-si, Gyeonggi-do, Korea
³CIST (Center for Information Security Technologies),
Korea University
Seoul, 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

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 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

* Corresponding author: Seungjoo Kim
This work was supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract UD100002KD.
This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the “ITRC” support program supervised by the NIPA(National IT Industry Promotion Agency)” (NIPA-2010-C1090-1001-0004)
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 ap
proved 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, “X” means the scheme does not guarantee the requirement, “△” means the scheme guarantee the requirement but it is inefficient, and “-“ means not applicable to the scheme.

Table 1. Time delay

<table>
<thead>
<tr>
<th>Key setup time (sec)</th>
<th>Certificate verification time (sec)</th>
<th>Data communication time (sec) by baud rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASKMA+</td>
<td>0.000021</td>
<td>38400, 19200, 9600, 4800, 2400, 1200, 600, 300, 110 (baud)</td>
</tr>
<tr>
<td>ASKMA</td>
<td>0.000021</td>
<td>38400, 19200, 9600, 4800, 2400, 1200, 600, 300, 110 (baud)</td>
</tr>
<tr>
<td>RSA</td>
<td>0.005714</td>
<td>38400, 19200, 9600, 4800, 2400, 1200, 600, 300, 110 (baud)</td>
</tr>
<tr>
<td>BD</td>
<td>0.007994</td>
<td>38400, 19200, 9600, 4800, 2400, 1200, 600, 300, 110 (baud)</td>
</tr>
<tr>
<td>TGDH</td>
<td>0.008514</td>
<td>38400, 19200, 9600, 4800, 2400, 1200, 600, 300, 110 (baud)</td>
</tr>
<tr>
<td>GDH</td>
<td>0.161040</td>
<td>38400, 19200, 9600, 4800, 2400, 1200, 600, 300, 110 (baud)</td>
</tr>
<tr>
<td>CKD</td>
<td>0.055184</td>
<td>38400, 19200, 9600, 4800, 2400, 1200, 600, 300, 110 (baud)</td>
</tr>
<tr>
<td>AGKA WNN</td>
<td>0.001644</td>
<td>38400, 19200, 9600, 4800, 2400, 1200, 600, 300, 110 (baud)</td>
</tr>
<tr>
<td>TT</td>
<td>0.008224</td>
<td>38400, 19200, 9600, 4800, 2400, 1200, 600, 300, 110 (baud)</td>
</tr>
<tr>
<td>NCKW</td>
<td>0.002030</td>
<td>38400, 19200, 9600, 4800, 2400, 1200, 600, 300, 110 (baud)</td>
</tr>
</tbody>
</table>

Table 2. Total time delay

<table>
<thead>
<tr>
<th>Total time delay (sec) by baud rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>115200 (baud)</td>
</tr>
<tr>
<td>ASKMA+</td>
</tr>
<tr>
<td>ASKMA</td>
</tr>
<tr>
<td>RSA</td>
</tr>
<tr>
<td>BD</td>
</tr>
<tr>
<td>TGDH</td>
</tr>
<tr>
<td>CKD</td>
</tr>
<tr>
<td>AGKA WNN</td>
</tr>
<tr>
<td>TT</td>
</tr>
<tr>
<td>NCKW</td>
</tr>
</tbody>
</table>

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 5.
Table 3. Security requirements

<table>
<thead>
<tr>
<th></th>
<th>ASKMA+</th>
<th>ASKMA</th>
<th>RSA</th>
<th>BD</th>
<th>TGDH</th>
<th>GDH</th>
<th>AGKA W/MN</th>
<th>CKD</th>
<th>TT</th>
<th>NCKW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broadcasting</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Multicasting</td>
<td>△</td>
<td>△</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Group Key Secrecy</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Forward Secrecy</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Backward Secrecy</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Perfect Forward Secrecy</td>
<td>-</td>
<td>-</td>
<td>×</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Key Freshness</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Availability</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

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.

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.
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 PROPOSED 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., \( GM = MT \cup RT \);
• \( RT \): \( RT = \{ RT_1, \ldots, RT_m \} \) is the set of RTU;
• \( MT \): \( MT = \{ MT_0, \ldots, MT_n \} \) 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 \( k \in \mathbb{N} \);
• \( q \): order of the algebraic group;
• \( r_i \): \( MT_i \)'s random number \( r_i \in \mathbb{Z}_q \);
• \( k_i \): \( MT_i \)'s intermediate key;
• \( K_{k,j} \): \( MT_k \)'s \( j \)th key at level \( i \) in a binary tree.
• \( E_g(r_i(K_g)) : (K_g)^{r_i} \mod p \)

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 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:

**Figure 2. Key hierarchy**

- \( RT_i \) 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

...
HYBRID KEY MANAGEMENT ARCHITECTURE FOR ROBUST SCADA SYSTEMS

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

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 $g^{r_0} \mod p$, 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 $g^{r_{m+1}} \mod p$ and sends it to the $MT_0$ with a digital signature.
• Step 3: The new device $MT_{m+1}$ and $MT_0$ compute $g^{r_{m+1}} \mod p$.
• Step 4: $MT_0$ checks the validity of the digital signatures, generates a new group key $K'_g$, computes $IK'_i = (K'_g)^{g^{r_i}} \mod p$ ($i \in [1,m+1]$), 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 $K'_g = K_g^{g^{r_i}} \mod p$.

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 $r_i$s, each group member cannot know the other group member’s $g^{r_i}$, since our protocol uses exponentials to compute $IK'$. It can be applied to a leave and replace protocol as well as a join protocol.
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 \( IK'_i = (K'_g)^{g^{0i}} \mod p \) (\( i \in \mathbb{Z} \))
HYBRID KEY MANAGEMENT ARCHITECTURE FOR ROBUST SCADA SYSTEMS

\[1, m\] and \(i \neq j\), and signs it.

- Step 2: \(M_{T_0}\) sends \(IK'_i (i \in [1, m] \text{ and } i \neq j)\) to \(MT_i\) with a digital signature.
- Step 3: Upon receipt of the message, each member \(MT_i (i \in [1, m] \text{ and } i \neq j)\) computes \(K'_g = g^{r_i r_j^i} \mod p\).

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.

### 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 \(M_{T_0}\) generates a new group key \(K'_g\), encrypts it with each device’s key \(g^{r_i} \mod p (i \in [1, m] \text{ and } i \neq n)\) and then sends it. In the second place, the SCADA system runs a join protocol, so \(M_{T_0}\) and the new \(MT'_n\) make a new key \(g^{r_n} \mod p\) and share the new group key \(K'_g\). The replace protocol runs as follows:

- Step 1: \(M_{T_0}\) generates a new group key \(K'_g\), computes \(IK'_i = (K'_g)^{g^{r_i}} \mod p (i \in [1, m] \text{ and } i \neq n)\), and signs it.
- Step 2: \(M_{T_0}\) sends \(IK'_i (i \in [1, m] \text{ and } i \neq n)\) to \(MT_i\) with a digital signature.
- Step 3: Upon receipt of the message, each member \(MT_i (i \in [1, m] \text{ and } i \neq n)\) computes \(K'_g = g^{r_i r_j^i} \mod p\).
- Step 4: \(M_{T_0}\) sends \(g^0 \mod p\) 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 \(g^{r_n} \mod p\) and sends it to the \(M_{T_0}\) with a digital signature.
- Step 6: \(MT'_n\) and \(M_{T_0}\) compute \(g^{r'_n} \mod p\).
- Step 7: \(M_{T_0}\) checks the validity of the digital signatures, generates a new group key \(K'_g\), computes \(IK'_n = (K'_g)^{g^{r'_n}} \mod p\), and signs it.
- Step 8: \(M_{T_0}\) sends \(IK'_n\) to \(MT'_n\) with a digital signature.
- Step 9: Upon receipt of the message, \(MT'_n\) computes \(K'_g = g^{r'_n r_j^n} \mod p\).

Fig. 5 shows a simple illustrative example of the leave protocol, where a broken device is \(MT_4\) and \(m = 4\).
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:

$$SK_u = H(K_{h,j}^k, TVP)$$ (1)

where $K_{h,j}^k$ 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:

$$SK_B = H(K_g, TVP)$$ (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 communi-
cation efficiency and security. So, we define the QoS function to find the period [23].

\[ QoS = CI + SI \] (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 \( \delta \) is the time delay caused by updating keys, and CI is computed below:

\[ CI = (T - \delta) / T \] (4)

Since the period to update the keys is in inverse proportion to the \( \delta \), we can modify the above formula:

\[ CI = (T - \delta) / T = (T - (k / tp)) / T \] (5)

where \( k \) is a constant and \( tp \) 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]:

\[ \frac{(\lambda l)^n}{n!} e^{-\lambda l}, \quad n=0,1,... \] (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 \( t = tp \).

\[ SI=e^{\lambda tp} \] (7)

In [23], \( \lambda \) 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

\[ SI=e^{G_t+\lambda_i+\lambda_{ni}}lp \] (8)

where \( \lambda_i \) is the mean of the number of successful attacks caused by logical errors, \( \lambda_i \) is the mean of the number of successful invasive attacks, and \( \lambda_{ni} \) 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.B. So, we can assign \( \lambda_i \) of our scheme to 0.

Finally, the QoS function can be expressed by \( tp \).

\[
QoS = \frac{T_{khp}}{T} + e^{(\lambda_i + \lambda_n) \lambda_p}
\]  

To maximize the QoS function, a differentiation of the QoS function at a \( tp \) should be 0.

\[
\frac{dQoS(tp)}{dp} = \frac{k}{T_{tp}^2} \left( (\lambda_i + \lambda_n) e^{(\lambda_i + \lambda_n) \lambda_p} \right) = 0
\]  

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

V. 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 MTs, 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.
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.

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

<table>
<thead>
<tr>
<th></th>
<th>SKE</th>
<th>SKMA</th>
<th>ASKMA</th>
<th>ASKMA+</th>
<th>Proposed Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTU</td>
<td>m(1+r)</td>
<td>m(1+r)</td>
<td>2m-l+mr</td>
<td>2m-l</td>
<td>m+2</td>
</tr>
<tr>
<td>Each SUB-MTU</td>
<td>1+r</td>
<td>1+r</td>
<td>r+1+log2m</td>
<td>2r+log2m</td>
<td>2r+1</td>
</tr>
<tr>
<td>Each RTU</td>
<td>1</td>
<td>1</td>
<td>2+log2m</td>
<td>1+log2r</td>
<td>1+log2r</td>
</tr>
</tbody>
</table>

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

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

![The number of multicasting target RTUs](image)

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

**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] and [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 theorem 1 and theorem 2 our scheme is secure.

VI. 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

10. Modbus, “MODBUS over serial line specification and implementation guide V1.02.”

Donghyun Choi received his B.E. degree in Electrical and Computer Engineering from Sungkyunkwan University, Korea, in 2005 and a 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 a M.S. degree in Computer Science from Sungkyunkwan University, Korea, in 2008. He is currently undertaking a Ph.D. course in Mobile Systems Engineering at Sungkyunkwan University. 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 have 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.