디지털산업정보학회 논문지 제5권 제2호-2009년 6월

PCS/SCADA 센서 네트워크용 키 관리 프로토콜에 대한 보안 분석

박 동 국

A Security Analysis of a Key Management Scheme for PCS/SCADA Sensor Networks[*]

Park, DongGook

〈Abstract〉

Process Control Systems (PCSs) or Supervisory Control and Data Acquisition (SCADA) systems have recently been added to the already wide collection of wireless sensor networks applications. The PCS/SCADA environment is somewhat more amenable to the use of heavy cryptographic mechanisms such as public key cryptography than other sensor application environments. The sensor nodes in the environment, however, are still open to devastating attacks such as node capture, which makes designing a secure key management challenging. Recently, Nilsson et al. proposed a key management scheme for PCS/SCADA, which was claimed to provide forward and backward secrecies. In this paper, we define four different types of adversaries or attackers in wireless sensor network environments in order to facilitate the evaluation of protocol strength. We then analyze Nilsson et al. 's protocol and show that it does not provide forward and backward secrecies against any type of adversary model.

Key Words : Wireless sensor network, Forward and backward secrecy, Key management, Process control systems, Supervisory control and data acquisition

Ⅰ. Introduction

Process Control Systems (PCSs) or Supervisory Control and Data Acquisition (SCADA) systems are used to monitor and control a plant or equipment in industries such as energy, oil and gas refining and transportation. These systems encompass the transfer of data between the network manager and a number of Remote Terminal Units (RTUs), sensor nodes, etc. A SCADA system gathers critical information (such as where a leak in a pipeline has occurred) and then transfers this information back to the network manager. The network manager is responsible for alerting the home station about the leak and carrying out necessary analysis such as determining whether the leak is critical or not.

The owners and operators of SCADA systems aim to increase the monitoring sensitivity of their systems

* This work was supported (in part) by Research Foundation of Engineering College, Sunchon National University.

and reduce the day to day running cost wherever it is possible. Due to the intelligent monitoring capabilities of the Wireless Sensor Networks (WSNs), integration between SCADA and WSNs can be one way to achieve these aims. WSNs facilitate the monitoring process by performing specific tasks such as sensing physical phenomena at a remote field and then reporting them back to the network manager. They can form the "eyes and ears" of SCADA systems. Nodes, which are capable of performing functions such as gas detection and temperature sensing, provide information that can tell an experienced operator how well oil/gas pipelines are performing.

Roman et al. highlighted the role that WSNs can play in SCADA [13]. They argued that WSNs can aid SCADA's functionalities by providing monitoring, alerts, and information on demand. However, vulnerabilities related to WSNs can be introduced to SCADA. One of those potential vulnerabilities is *node capture* by an adversary, which leads to the security compromise of sensor nodes given the lack of tamper resistance packaging [4].

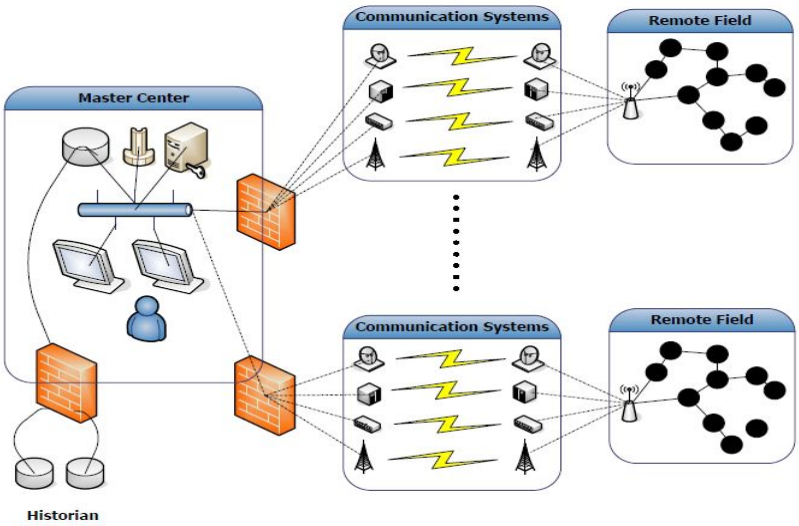
With limited resources in sensor nodes, defeating this type of threat is very hard. Node capture will translate into compromise of all the credentials stored in the sensor node. Furthermore, the adversary can compromise all the software codes installed within the sensor node, especially random number generation functions. For example, he can modify the codes or replace them with his own codes to mislead functions related to SCADA/PCS, or use a fixed number for random numbers for input to security protocols. In this regard, the question arises: Can we protect the *past* and *future* keys from the adversary even after he gains all the *current* secret keys stored in the captured nodes? A key management protocol is said to provide "backward secrecy" and "forward secrecy" when its answer to the above question is positive.

To the best of our knowledge, however, there is not yet a single key management protocol for WSNs which is both backward secure and forward secure. Recently, Nilsson et al. proposed a key management scheme for WSNs in PCS/SCADA environments. They claim that their scheme is both forward and backward secure. To be fair, we classify adversaries into four different types according to their capabilities, and analyzed Nilsson et al. 's scheme to veryify if it provides forward and backward secrecies against each type of adversaries.

II. SCADA

To best understand the scheme proposed by Nilsson et al., some understanding of SCADA is in order. Today's SCADA systems (the third generation) are a combination of legacy and modern technology [8]. It has become an open system architecture rather than a vendor controlled architecture as in the second generation of SCADA. It uses open standards and protocols which facilitate distribution of the functionalities of SCADA. We refer the readers interested in the differences between these generations to the paper by McClanahan [8]. Figure 1 shows a simplified SCADA system architecture which is composed of the following components:

Master Center. The master center component contains the network manager, human machine interaction, database storage, processing server, etc. It has the highest physical security level compared to other components. Generally speaking, it receives



<Figure 1> The simplied version of PCS/SCADA

monitoring information from remote fields (through the communication system component), processes it, and then makes decisions.

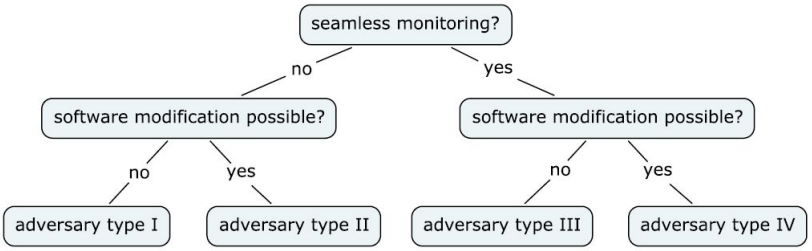
Historian. The historian is a backup for the SCADA system data which is often located in a separate subnet different to the one where the master center component exists. The master center component is able to access the historian in order to backup the data of the SCADA system.

Remote Fields. The remote fields are composed of substations (gateways) and intelligent electronic devices (IEDs) [1] which can be physically distant from the SCADA master center and in many cases are not physically secured due to the largeness or remoteness of the coverage area. The substation connects IEDs with the master center component through the communication component. It has a high degree of complexity and might have better physical security than IEDs. The IEDs can be sensor nodes, remote terminal units, or relays to name a few.

Communication System. The communication systems are responsible for transferring monitored data (control data) from remote field components (master center) to master center component (remote field components). This communication can be done via fiber optics, radio, satellite, etc.

III. Adversary Model and Security Concerns

Our main interest and motive in this work was to find a key management scheme which is resilient to node capture: i. e., a scheme that enables sensor nodes to *recover* its secure status even after they have been captured and then released back. Consequently, we are interested in what the adversary can do when a node is captured, and after it is released back. Key disclosure is a trivial fact; what else should be done by the adversary to keep control of the node after he put it back to the field? He will hope that the node uses



<Figure 2> Classification of Adversaries: "Seamless monitoring" means the adversary keeps monitoring every subsequent key update message after compromising a sensor node; "software modication" includes alteration of any software installed in the node, especially the random number generator.

values of his choice for all cryptographic keys or keying materials. For this purpose, he may try to modify software components (especially the random number generation part), and monitor all or part of the subsequent key update messages. In this regard, we use the following criteria to classify the adversaries.

- The adversary can read and modify all the software codes and configurations, including secret keys, installed in the sensor node.
- The adversary can carry out seamless monitoring of all the subsequent key update protocol exchanges.

According to the above two criteria, we divide the adversaries into four distinct types as shown in Figure 2. Type I is the weakest adversary: neither seamless monitoring nor software compromise; Type IV is the strongest: seamless monitoring and software compromise. Type IV is so much powerful that it is unlikely to devise any practical cryptographic countermeasure for WSNs. The use of tamper-proof technology will be needed to cope with this type of adversary, but it is outside the scope of this paper.

One interesting point here is that the assumption of software modification is equivalent to that of software-based random number generation, in terms of their consequence in the context of cryptographic protocols. Software algorithm-based random number generation does not give true random numbers, which can only be obtained from a strong physical source of randomness. One consequence of this equivalence is that it makes no sense to use expensive tamper-proof technologies while true random number generation not used. Put a different way, we do not have to bother with true random number generation when software modification is assumed to be an easy work for the adversary.

Having identified different types of adversaries, we have the following concerns with regard to node capture and the consequent disclosure of all the internal data of the captured node:

- Past key secrecy: The past keys should not be compromised.
- Future key secrecy: The future keys should not be compromised.

The requirement of resilience to node capture rules out the use of any long-term keys; the keys must change or evolve continuously over time, with old

prior keys deleted securely. In other words, we require a key evolution scheme in order to achieve past/future key secrecy against the threat of node capture.

Terminology. To the best of our knowledge, the terms "past/future key secrecy" have never been used in previous literature. Similar terminology include "(perfect) forward secrecy" and "backward secrecy", which has always been quite confusing. The term "(perfect) forward secrecy" goes back to Günther [3]. The original term assumes a long-term key and session keys established by the key, and means that the current session key is not compromised by the "future" (thus, the expression "forward") exposure of the long-term key. This terminology, somehow, seems to have got a slightly different usage in the context of group key communication; it concerns about the contamination of a group key at a particular time by the compromise of an older/newer group key. The inherent ambiguity has brought a twin terminology: "backward secrecy". Some authors choose the term "backward secrecy" to mean "forward secrecy" called by other authors, and vice versa. To avoid all this confusion, we will use a new more concrete expression: "past/future key secrecy".

IV. The Key Management Scheme by Nilsson et al.

There are several papers dealing with key management designs for SCADA systems such as [2, 11]. However, these designs either use heavy cryptographic mechanisms, which do not suit resource constrained devices, or do not consider the integration of WSNs within SCADA.

To the best of our knowledge, the only existing key management in WSNs for SCADA/PCS environment has been proposed by Nilsson et al.[9]. They designed two key update protocols: the first one updates the pairwise symmetric key between the network manager M and a sensor node N (as described in Protocol 2) while the other scheme updates the global or group key among M and the whole group of sensor nodes (as described in Protocol 1). They claimed that the protocols provide both forward and backward secrecy (or in our newly defined terminology, they provide both past and future key secrecy). It is unfortunately not the case.

In Table 1, we summarize the notations used for the description of Nilsson et al. 's protocols.

M	Network manager
N	Sensor node
K_G	Goup key
K_{MN}	Pairwise key between M and N
r_X	Random nonce chosen by entity X
K_M^{-1}, K_M	Asymmetric key pair of network manager
$\{m\}_K$	Message m encrypted under key K
$MAC_K(m)$	Message authentication code of m using key K

<Table 1> Notations for protocol description

Protocol 1 Group key update protocol by Nilsson et al.

M: generates a new group key $K_G^{'}$ and a random number r_M
1. $M \rightarrow N$: $\{K_G^{'}, r_M\}_{K_{MN}}$
2. $M \leftarrow N$: $MAC_{K_G^{'}}(N, r_M)$

Protocol 2 Pairwise key update protocol by Nilsson et al.

N: generates a random number r_N
1. $M \leftarrow N$: $\{r_N\}_{K_M}, MAC_{K_{MN}}(\{r_N\}_{K_M})$
M, N: compute the new pairwise key $K_{MN}^{'} = h(K_{MN}, r_N)$

To initiate the group key update protocol (Protocol 1), *M* generates a new group key, $K_G^{'}$, randomly. It then encrypts it together with another random number, r_M, using K_{MN} as the encryption key, and sends the ciphertext over the network to the target group. No node in the group has any clue whether the received key is fresh or not. In other words, the freshness property, from the viewpoint of *N* does not hold since the two values (the new group key $K_G^{'}$ and the random number r_M) are random values chosen by *M*. It is both impractical and insecure for each sensor node to maintain a list of keys that have been used. Thus, an external adversary will be able to record a rekeying message and then re-inject it into the network, which leads to updating the group key with an old key. Consequently, the group enters a key mismatch phase where the key version that the group of sensors uses and what *M* has are different.

One good security practice is to minimize the damage caused by a compromised node. However, the authors did not consider common attacks in WSNs that an adversary is capable of launching attacks such as selective forwarding [5] or node compromise [4]. If a single sensor node has the ability to affect the operation of a good number of sensor nodes, then the adversary will try to compromise that node. For example, if an adversary compromised a particular sensor node in a multi-hop path, then it would be able to enforce all other nodes downstream to enter the key mismatch phase. The adversary simply drops the rekeying message from *M* for the group key, and then use the new group key to calculate MACs on their identities and the received nonce, which results in a successful impersonation attack. We can easily fix the problem by replacing the MAC data with another one: e. g., $MAC_{K_{MN}}(K_G^{'}, r_M)$.

Moreover, to initiate the pairwise key update protocol (Protocol 2), *N* generates a random number, r_N, and encrypts it with *M*'s public encryption key K_M. It subsequently computes the *MAC* on the encryption result and sends this *MAC* and the encryption result over the network to *M*. The new pairwise key can be calculated, at the sender *N* and at the receiver *M*, by hashing r_N with the previous pairwise key. This means that the new pairwise key is always determined by *N*. The adversary, consequently, is able to know all the future keys once he compromised *N*.

A closer look at the protocols, Protocol 1 and Protocol 2, reveals more serious defects of them.

Defect I. The whole value of the new group key are directly carried by the protocol messages, encrypted under the pairwise key K_{MN}. The consequence is that compromise of the pairwise key for just one node leads to compromise of the group key for the whole group. This is a more serious problem than it might appear, because the pairwise key compromise does not necessarily require node capture.

Defect II. The value of the new pairwise key $K_{MN}^{'}$ is only determined by the sensor node. When the adversary of Type II or IV (he can compromise the key generation codes stored in the node) captures the node, all the future pairwise keys for the node can be pre-determined by the adversaries. Namely, physical compromise of the node immediately leads to compromise of all the future pairwise keys if the adversary can modify the codes installed in the node. This, in turn, leads to compromise of all the future group keys as well because, as mentioned in Defect I, the group key is delivered encrypted under the

pairwise key. Hence, contrary to their claim, the scheme does not provide "future key secrecy", against node compromise, for either the pairwise key or the group key.

Defect III. Although not explicitly shown in the protocol descriptions above, the key input for the new pairwise key K'_{MN} is not really random in their scheme; it is in fact a function of a pre-installed secret key and a counter value stored in the node. This means that, when the node is captured and all the installed data including keys exposed to the adversary, all the past pairwise keys as well as the future keys can immediately be computed even without recording a single key update message! In fact, this disaster is not just because of Defect III, but also due to Defect II.

Note that, due to Defect III combined with Defect II, the adversary does not have to modify the node's software at all in order to extract all the past and future pairwise keys. Hence no minimum level of past or future key secrecy against node compromise in their scheme. Moreover, the adversary can extract any group key in the past or future if he has got the records of the corresponding group key update message. Note also that, for this, "seamless" monitoring is not needed by the adversary. What does this mean? The scheme is, in terms of either kind of key, neither forward nor backward secure against node compromise for all the types of adversary I, II, III and IV (see Figure 2).

As for past key secrecy, we note two proposed schemes in the WSN context: Klonowski et al. [6] and Mauw et al. [7]. Both schemes use hash functions in order to achieve key evolution. Both schemes, however, are intended to be used not for group key update but for updating pairwise keys for node-to-node [6, 12] or node-to-base station communication [7].

On the other hand, as for future key secrecy, Mauw et al. 's protocol does not provide this property. The protocol is based on a hash chain scheme originally proposed for RFID security [10]. In RFID environments, protecting secret tag information from tampering in the future is a big concern while it does not seem to be such a prime concern in WSNs. This is because it is more authentication and integrity than privacy that really matters in WSNs, especially SCADA/PCS. Hence, future key secrecy is more valued than past key secrecy. On the other hand, the protocol proposed by Klonowski provides future key secrecy in a "weak" sense; namely, it will be computationally hard for the adversary to compute a future key from the current compromised key if he fails to record, say ten, subsequent evolution steps [12].

Ⅴ. Conclusion and Future Work

WSNs has brought devastating security threat: node capture. The threat is so powerful that almost all existing key management protocols are just helpless because it overthrows the fundamental assumption for cryptographic system design: long term secret keys are securely stored. This is why so called forward secrecy and backward secrecy are required in cryptographic key management protocols for WSNs. Both terminologies are rather misleading and confusing, and so we propose more proper ones: future key secrecy and past key secrecy.

Nilsson et al. [9] have recently proposed a key management scheme for WSN applications in PCS/SCADA environments, which was incorrectly

claimed to provide future and past key secrecies. Some proposals (only for pairwise key update) provide past key secrecy, but not future key secrecy [7, 6].

We noticed that any cryptographic countermeasure alone cannot prevent the most powerful adversary in the WSN context; he can capture a node to extract all the confidentials, modify any built-in codes, and seamlessly monitor to keep control of the node. This kind of attackers can only be fought by using tamper-proof technologies as well as cryptographic ones. The assumption regarding this type of adversaries, however, is by no means the most usual or reasonable assumption. Seamless monitoring requires the adversary not to lose every single session for group key or pairwise key update. The task of modification of random number generation codes will add another burden to that.

In order to measure the resilience of key management protocols, we derived four different types Ⅰ, Ⅱ, Ⅲ, and Ⅳ of adversaries varying in their capability with regard to seamless monitoring and software manipulation. As shown in Section Ⅲ, Nilsson et al. 's scheme, contrary to their claims, turned out to provide neither past key secrecy nor future key secrecy against node compromise by any type of adversaries. This result is rather surprising because they adopted public-key encryption to build the pairwise key update protocol, which is still arguably expensive for capability-limited environments like WSNs.

Our analysis shows that equipping a key management protocol with forward secrecy and backward secrecy (or past key secrecy and future key secrecy in our own terms) is not a trivial work. As future work we pan to design a new protocol providing both properties, and expect to report on that in a subsequent paper.

References

[1] C. Beaver, D. Gallup, W. Neumann, and M. Torgerson, "Key Management for SCADA", Technical Report SAND2001-3252, Sandia National Laboratories-Cryptography and Information Systems Surety Department, March 2002.

[2] R. Dawson and C. Boyd and E. Dawson and J. G. Nieto, "SKMA: a Key Management Architecture for SCADA Systems", ACSW Frontiers 2006, pp. 183-192.

[3] C. Gunther, "An Identity-Based Key-Exchange Protocol", Advances in Cryptology-Eurocrypt'89, Springer-Verlag (LNCS 434), 1990, pp. 29-37.

[4] C. Hartung and J. Balasalle and R. Han, "Node Compromise in Sensor Networks: The Need for Secure Systems", Technical Report CU-CS-990-05, Department of Computer Science, University of Colorado at Boulder, January 2005.

[5] C. Karlof and D. Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures", First IEEE International Workshop on Sensor Network Protocols and Applications, 2003, pp. 293-315

[6] M. Klonowski and M. Kutylowski and M. Ren and K. Rybarczyk, "Forward-Secure Key Evolution in Wireless Sensor Networks", CANS, Springer-Verlag (LNCS 4856), 2007, pp. 102-120.

[7] S. Mauw, I. van Vessem, and B. Bos, "Forward Secure Communication in Wireless Sensor Networks", Third International Conference

Security in Pervasive Computing (SPC'06), Springer-Verlag (LNCS 3934), 2006, pp. 32-42.

[8] R. McClanahan, "SCADA and IP: Is Network Convergence Really Here?", Industry Applications Magazine, IEEE, 2003, pp. 29-36.

[9] D. K. Nilsson, T. Roosta, U. Lindqvist and A. Valdes, "Key Management and Secure Software Updates in Wireless Process Control Environments", Proceedings of the first ACM conference on Wireless network security (WiSec '08), March 31-April 2, 2008, Alexandria, VA, pp. 100-108.

[10] M. Ohkubo, K. Suzuki, and S. Kinoshita, "Cryptographic Approach to Privacy-Friendly Tags", RFID Privacy Workshop, 2003.

[11] L. Pietre-Cambacedes and P. Sitbon, "Cryptographic Key Management for SCADA Systems-Issues and Perspectives", International Journal of Security and its Applications, Vol. 2, No. 3, 2008, pp. 31-40.

[12] M. Ren, K. D. Tanmoy and J. Zhou, "Diverging Keys in Wireless Sensor Networks", Information Security, Springer-Verlag (LNCS 4176), 2006, pp. 257-269.

[13] R. Roman and C. Alcaraz and J. Lopez, "The Role of Wireless Sensor Networks in the Area of Critical Information Infrastructure Protection", Information Security Technical Report, Vol. 12, Issue 1, 2007, pp. 24-31.

■ 저자소개 ■

박 동 국
Park, DongGook

2004년 2월~현재
순천대학교 정보통신공학부 조교수
1989년 3월~2004년 2월
KT 수석연구원
2001년 9월 호주 QUT 데이터통신과 (공학박사)
1989년 2월 KAIST 전기/전자공학과 (공학석사)
1986년 2월 경북대학교 전자공학과 (공학사)

관심분야 : 암호 프로토콜 모델링 및 분석
E-mail : dgpark6@sunchon.ac.kr

논문접수일 : 2009년 5월 7일
수 정 일 : 2009년 5월 25일
게재확정일 : 2009년 5월 30일