Improving a Forward & Backward Secure Key Management Scheme for Wireless Sensor Networks

DongGook Park, *Member, KIMICS*

***Abstract*—**Park proposed a forward & backward Secure key management scheme in wireless sensor networks for Process Control Systems (PCSs) or Supervisory Control and Data Acquisition (SCADA) systems [7]. The scheme, however, is still vulnerable to an attack called "sandwich attack": two nodes captured at times t_1 and t_2, respectively, surrenders all the group keys used between times t_1 and t_2. In this paper, we propose a fix to the scheme, which can limit the vulnerable time duration to an arbitrarily chosen time span while keeping the forward and backward secrecy of the scheme untouched.

Index Terms—sandwich attack, node capture, wireless sensor network, forward and backward secrecy, key management, process control systems, supervisory control and data acquisition.

I. INTRODUCTION

Wireless sensor networks (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 **[7]**, and Park proposed more proper ones: future key secrecy and past key secrecy.

- **Past key secrecy:** the past keys should not be compromised even when the current key is compromised.
- **Future key secrecy:** The future keys should not be compromised even when the current key is compromised.

Usually WSNs employs two types of key: (1) the group key shared among all the sensor nodes and the network manager, and (2) the pairwise key shared between each node and the network manager.

Nilsson et al. [6] have recently proposed a key management scheme for WSN applications in PCS/SCADA environments [1],[2],[5],[8], which was incorrectly claimed to provide future and past key secrecies. The scheme uses key transport, not key agreement, where the new pairwise key is determined solely by the sensor node. This design flaw has brought a vital flaw to their scheme with regard to node capture attacks. The attacker, after capturing a node, can choose his own values for future pairwise keys.

Some proposals (only for pairwise key update) provide past key secrecy, but not future key secrecy [4],[3].

Park proposed a key management scheme for PCS/SCADA environments, which provides both past key secrecy and future key secrecy **[7]**. The scheme applied Lamport's reverse hash chain as well as usual hash chain to provide both future and past key secrecies. The scheme avoids the delivery of the whole value of a new group key for group key update; instead only the half of the value is transmitted from the network manager to the sensor nodes. This way, the compromise of a pairwise key alone does not lead to the compromise of the group key, which was not the case in the scheme by Nilsson et al.

Unfortunately, however, Park's scheme is vulnerable to a new kind of attack which is called "sandwich attack" in **[7]**: two nodes captured at times t_1 and t_2, respectively, surrenders all the group keys used between times t_1 and t_2.

We propose a simple fix to Park's scheme which does not affect its desirable features and significantly mitigates the weakness with regard to sandwich attacks.

Manuscript received September 28, 2009 ; revised November 13, 2009.

DongGook Park is with the Department of Information Technology, SunChon National University, SunChon, JeonLaNamDo, 540-742, Korea (Tel: +82-61-750-3597, Fax: +82-61-750-3590, Email: dgpark6@sunchon.ac.kr)

II. PARK'S PROTOCOOL

In this section, we briefly describe Park's protocol. Table 1 shows the notation as used in the description of Park's protocol in [7].

Table 1 Notations used in this paper

M	Network manager
N	Sensor node
K_{MN}	Shared pairwise key between M and N
s_0, t_0	Pre-installed global secret data in every N
K_G^i	The *i*-th group key ($i \geq 0$)
r_X	Random nonce chosen by entity X
(K_M^{-1}, K_M)	Asymmetric key pair of network manager
$\{m\}_K$	Encryption of message *m* under the key K
$h(\cdot)$	A cryptographic hash function
$h_K(\cdot)$	A keyed hash function using the key K

In order to fight node capture attacks, Park's protocol implemented a novel key evolution, which combines a forward hash chain and a reverse hash chain to provide both past key secrecy and future key secrecy at the same time. The network manager chooses two secret data, s_0 and t. It then prepares two hash chains of length *n*: the forward hash chain starts from s_0 and ends at $h^{n-1}(s_0)$ by applying the hash function $n-1$ times; the reverse hash chain starts from t and ends at t_0 by repeating the hash operation $n-1$ times.

$$s_0,\ h^1(s_0) := h(s_0),\ \ldots\ ,\ h^{n-1}(s_0) := h(h^{n-2}(s_0))$$
$$t_0 := h(h^{-1}(t_0)),\ h^{-1}(t_0) := h(h^{-2}(t_0)),\ \ldots\ ,\ h^{-(n-1)}(t_0) := t$$

Note that $h^{-i}(t_0)$ means the *i*-th preimage of t_0 in the reverse hash chain, hence $t_0 = h^{n-1}(t)$. Now, the network manger pre-installs s_0 and t_0 into the sensor nodes before deployment.

When deployed, the sensor nodes updates the group key using Protocol 1 as shown below.

Protocol 1: The protocol for group key update

1. $M \rightarrow N$: $i, \{h^{-i}(t_0)\}_{K_{MN}}$
2. $M \leftarrow N$: $h_{K_{MN}}(K_G^i)$

M, *N*: increment the group key index from $i-1$ to i, and update the value of the group key (i.e., $K_G^i := h^i(s_0) \oplus h^{-i}(t_0)$).

Whenever the protocol is executed, the group key is updated from K_G^{i-1} to K_G^i. The network manager *M* releases the *i*-th preimage encrypted under the pairwise key K_{MN}. The sensor node, upon receipt of the first protocol message, validates the preimage by checking if $h(h^{-i}(t_0)) = h^{-(i-1)}(t_0)$, and then hashes $h^{i-1}(s_0)$ to get $h^i(s_0)$. Finally, it computes the new group key K_G^i by computing $h^i(s_0) \oplus h^{-i}(t_0)$. The sensor node computes the keyed hash of the new group key and returns it to *M*, and stores $h^i(s_0)$ and $h^{-i}(t_0)$ for future use.

By combining two hash chains, the protocol provides a few nice features. First, the presence of the *i*-th preimage of t_0 in the first message assures the sensor nodes as to the authenticity of the message: i.e., the message could have been generated only by the network manager. Moreover, the message is fresh because the preimage has never been disclosed before.

Second, the protocol provides a sort of "self-synchronization" to the whole WSN system. Due to lossy transmission in WSNs, some sensor nodes may fall behind with group key update. The sensor nodes, however, will soon be able to catch up at the next rekeying: it can compute the correct value of the new group key simply by checking the difference of two index values – the received and the stored – and applying the corresponding number of hash operations.

Last, but most importantly, the protocol provides both forward key secrecy and backward key secrecy against two kinds of compromise: group key compromise and pairwise key compromise. Only with knowledge of either the group key K_G^i or the pairwise key K_{MN} alone, the attacker cannot compute the next group key K_G^{i+1} because it requires knowledge of $h^{i+1}(s_0)$ and $h^{-(i+1)}(t_0)$. Even when a node is captured and thus surrendering all the secret data including $h^i(s_0)$, $h^{-i}(t_0)$, K_G^i, and K_{MN}, the attacker cannot compute any past or future group/pairwise keys. In order to get the next group key, the attacker has to wait and monitor the next protocol run. Once he misses one instance of the protocol run for group key update, and subsequently if Protocol 2, as shown below, is executed, he loses the control of the sensor nodes entirely with regard to group keys and pairwise keys.

Even if he can seamlessly monitor all the protocol runs for group/pairwise key update, the attacker cannot get the new group/pairwise keys after the execution of Protocol 2 provided that he has no ability to modify the softwares embedded in the sensor node.

Protocol 2: The protocol for pairwise key update

1. $M \rightarrow N$: $i, \{h^{-i}(t_0), g^{r_M}\}_{K_G^{i-1}}$ # broadcast message
2. $M \leftarrow N$: $\{g^{r_N}\}_{K_{MN}}, h_{K_{MN}}(g^{r_M}, g^{r_N})$

M, *N*: increment the group key index from $i-1$ to i, and update the values of the pairwise key (i.e., to $K_{MN} := g^{r_M r_N}$) and the group key (i.e., to $K_G^i := h^i(s_0) \oplus h^{-i}(t_0)$)

Protocol 2 is intended to update both pairwise and group key updates. Use of Diffie-Hellman for pairwise key generation provides future/past key secrecies for pairwise keys. Note that the compromise of all the security data does not lead to compromise of the old/new pairwise keys because of the perfect forward secrecy enabled by Diffie-Hellman. The only way for the attacker to keep control of the future pairwise keys is modification of the software codes in the sensor node which generates the Diffie-Hellman component. Even in that case, he cannot predict the future pairwise key because of the key agreement property of Diffie-Hellman key exchange. Therefore, he must not fail in his attempt to monitor all the group/pairwise key update protocol executions.

Park derived an attacker model [7], where attackers are divided into four types according to their abilities: (1) seamless monitoring and (2) software modification. Park's scheme provides forward/backward secrecies against all types of attackers except the most powerful one (called Type IV in [7]) which can afford both conditions (1) and (2) above. In fact, Type IV attacker defeats any countermeasure by cryptography alone. For detailed description of the protocols and their features, we refer the readers to [7].

III. SANDWICH ATTACK AND A COUNTERMEASURE

As already mentioned in [7], Park's scheme suffers from a new kind of attack called "sandwich attack". Two nodes captured at times *i* and *j* (*i* < *j*), respectively, surrenders all the group keys used between times *i* and *j*. Here *i* and *j* are discrete time indices, which are intended to mean the group key indices as used in Protocols 1 and 2. The attacker captures a node at time *i*, compromising $h^i(s_0)$ and $h^{-i}(t_0)$. Thus he can compute all the subsequent hash images of the forward hash chain: $h^{i+1}(s_0)$, …, $h^{j-1}(s_0)$, $h^j(s_0)$. When he captures another node at time *j*, he can compute all the pre-images of the reverse hash chain: $h^{-j}(t_0)$, $h^{-(j-1)}(t_0)$, …, $h^{-(i+1)}(t_0)$. Now the attacker can compute all the group keys from *i* to *j* by the computation: $K_G^k := h^k(s_0) \oplus h^{-k}(t_0)$, where $i \le k \le j$.

This weakness comes from the design feature of the scheme: the combination of a forward hash chain and a backward hash chain. Our solution to this problem is simple: Break the reverse hash chain into shorter ones while not leaving any vulnerable security crack between their connection. The following protocols are a modified version of Protocols 1 and 2 to accommodate this idea.

Protocol 3: Protocol 1 equipped with the countermeasure against sandwich attacks

1. $M \rightarrow N$: $i, \{h^{-i}(t_0), \boxed{t_0'}\}_{K_{MN}}$
2. $M \leftarrow N$: $h_{K_{MN}}(K_G^i)$

M, *N*: increment the group key index from $i-1$ to i, set $h^{-i}(t_0) := t_0'$, and update the value of the group key (i.e., $K_G^i := h^i(s_0) \oplus h^{-i}(t_0)$),

Protocol 4: Protocol 2 equipped with the countermeasure against sandwich attacks

1. $M \rightarrow N$: $i, \{h^{-i}(t_0), \boxed{h^{-1}(t_0')}, g^{r_M}\}_{K_G^{i-1}}$ # broadcast message
2. $M \leftarrow N$: $\{g^{r_N}\}_{K_{MN}}, h_{K_{MN}}(g^{r_M}, g^{r_N})$

M, *N*: increment the group key index from $i-1$ to i, *reset the value of* $h^{-i}(t_0)$ *to* $h^{-1}(t_0')$, and update the values of the pairwise key (i.e., to $K_{MN} := g^{r_M r_N}$) and the group key (i.e., to $K_G^i := h^i(s_0) \oplus h^{-i}(t_0)$)

The protocol messages of Protocols 4 and 3 are exactly the same as those of Protocols 2 and 1 except for the addition of a new data $h^{-1}(t_0')$ and its commitment t_0', respectively. This addition enables the network manager *M* to restart a new reverse hash chain by choosing a new starting value t', and then computing successive hash images of t'. The final value of the hash chain is assigned to t_0'. Namely, *M* re-establishes the reverse hash chain.

It should be noted that, after the execution of Protocol 4, $h^{-i}(t_0)$ is no longer related to t_0 and thus $h^{-(i-1)}(t_0)$ as well; in fact, it has been reset to the value of $h^{-1}(t_0')$, i.e., $h^{-i}(t_0) := h^{-1}(t_0')$. It is just for notational convenience that we keep using the name $h^{-i}(t_0)$.

Inclusion of t_0' together with $h^{-i}(t_0)$ in the first message of Protocol 3 convinces the sensor node that t_0' has been originated from the network manager. Note that t_0' delivered to the sensor node encrypted under the pairwise key K_{MN}, not under the group key.

And then, it's keyed hash is returned to the network manager. Thus, any intermediate sensor node cannot modify t_0' without being detected. When $h^{-1}(t_0')$ is delivered to the sensor node *N* by the first message of Protocol 4, *N* can validate it by checking if $h(h^{-1}(t_0')) = t_0'$. Note that inclusion of the commitment data t_0' in the first message of Protocol 3 is essential: without it, any intermediate sensor node could modify $h^{-1}(t_0')$ because it is encrypted under the group key.

Interestingly, the modified protocols equipped with the countermeasure comes with a nice feature: re-establishing the reverse hash chain. With this feature, the sensor nodes doesn't have to be recollected to refill the reverse hash chain. Now, the network manager can initiate Protocol 4 to update the pairwise key and refill the reverse hash chain after several group key updates using Protocol 3. This way, the time span within which sandwich attacks succeed can be arbitrarily limited.

IV. CONCLUSION

Park's key management scheme has nice features: forward and backward secrecies or future/past key secrecies as called in [7]. Its design idea is the combination of forward and reverse hash chains for key evolution. The combination, however, has brought a new problem: vulnerability to sandwich attacks which is a special kind of node capture attack. Another potential issue with the scheme was how to refill the reverse hash chain.

We took advantage of message authentication feature of the first message of Protocol 1 of Park's scheme to deliver a commitment data from the network manager to the sensor nodes (as shown in Protocol 3). A subsequent execution of Protocol 2 was exploited to restart the reverse hash chain (as shown in Protocol 4). Thus, the sandwich attack can only be attempted to a very limited time interval which corresponds to the life time of a particular reverse hash chain.

In short, our countermeasure against sandwich attacks makes Park's scheme more resilient against node capture attacks and reinitialize the reverse hash chain at the same time.

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] 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.

[4] 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.

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

[6] 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.

[7] D. Park, "A Forward & Backward Secure Key Management in Wireless Sensor Networks for PCS/SCADA", *Journal of The Korea Society of Digital Industry & Information Management*, Vol.3, No.2, June, 2009, pp. 98-106.

[8] 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.

DongGook Park

Received B.S. degree from Kyungpook National University in 1986, and M.S. degree from the Korea Advanced Institute of Science and Technology (KAIST) in 1989, and Ph.D. degree from Queensland University of Technology (QUT), Brisbane, Australia in 2001. From 1989 to 2004, he worked for Korea Telecom (KT), where he was involved in a collaborative research between QUT and KT for future mobile communications security for three years. In 2004, he joined the Department of Information Technology, SunChon National University, Korea. His research interest includes modeling and analysis of authentication and key establishment protocols.