Mitigating Sandwich Attacks against a Secure Key Management Scheme in Wireless Sensor Networks for PCS/SCADA

Hani Alzaid and DongGook Park and Juan González Nieto and Ernest Foo

April 28, 2010

Abstract

Alzaid et al. 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. The scheme, however, is still vulnerable to an attack called the sandwich attack that can be launched when the adversary captures two sensor nodes at times t_1 and t_2, and then reveals all the group keys used between times t_1 and t_2. In this paper, a fix to the scheme is proposed in order to limit the vulnerable time duration to an arbitrarily chosen time span while keeping the forward and backward secrecy of the scheme untouched. Then, the performance analysis for our proposal, Alzaid et al.'s scheme, and Nilsson et al.'s scheme is given.

Keywords: sandwich attack; wireless sensor network; forward and backward secrecy; key management; process control systems; supervisory control and data acquisition; node

1 Introduction

Wireless sensor networks (WSNs) have a devastating security threat: node capture [9]. The threat is so powerful that almost all existing key management protocols are helpless because it overthrows the fundamental assumption for cryptographic system design which 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, in the current literature, are rather misleading and confusing. Therefore, Alzaid et al. proposed: future key secrecy and past key secrecy [3].

- **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 employ 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 sensor node and the network manager. Nilsson et al. [15] proposed a key management scheme for WSN applications in PCS/SCADA environments [4,7,14,16], which was incorrectly claimed to provide future and past key secrecy. Nilsson et al.'s scheme uses key transport, not key agreement, where the new pairwise key is determined solely by the sensor node. This 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 [12, 13]. Alzaid et al. have recently proposed a key management scheme for PCS/SCADA environments, which provides both past key secrecy and future key secrecy [3]. The scheme applied Lamport's reverse hash chain as well as usual hash chain to provide both future and past key secrecy. The scheme avoids the delivery of the whole value of a new group key for group key update; instead only 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 proposed by Nilsson et al. However, Alzaid et al.'s scheme is vulnerable to

a new kind of attack which is called the sandwich attack. This attack can be launched when the adversary captures two nodes at times t_1 and t_2, and then reveals all the group keys used between times t_1 and t_2. We propose a simple fix to Alzaid et al.'s scheme which does not affect its desirable features and significantly mitigates the weakness with regard to sandwich attacks.

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
$\{m\}_X$	Encryption of message m under the key K
$h(.)$	A cryptographic hash function
$h_K(.)$	A keyed hash function using the key K

2 Alzaid et al.'s Scheme

In this section, we briefly describe Alzaid et al.'s scheme. Table I shows the notations used throughout their paper. In order to fight node capture attacks, Alzaid et al.'s scheme 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_0. 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.

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

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 manager pre-installs s_0 and t_0 into the sensor nodes before deployment. When deployed, the sensor nodes update the group key as shown in Protocol 1.

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 ($K_G^i := h^i(s_0) \oplus h^{-i}(t_0)$).

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 reception 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 of the message's authenticity. In other words,

it assures that the network manager is the only one who generated the message. 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 revealing 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 the adversary 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, especially if he is not able to modify the software embedded in the sensor node such as random number generators. Protocol 2 is intended to update both pairwise and group keys. Use of Diffie-Hellman

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 and the group key ($K_{MN} := g^{r_M r_N}$, $K_G^i := h^i(s_0) \oplus h^{-i}(t_0)$).

for pairwise key generation provides future/past key secrecy 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 forward secrecy enabled by Diffie-Hellman. The only way for the attacker to keep control of the future pairwise keys is by modifying the software 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, the adversary must not fail in his attempt to monitor all the group/pairwise key update protocol executions. Alzaid et al. derived an attacker model, where attackers are divided into four types according to their abilities to perform: (1) seamless monitoring and (2) software modification [3]. Their scheme provides forward/backward secrecy against all types of attackers except the most powerful one (called Type IV), 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 [3].

3 Sandwich Attack and Countermeasure

As already mentioned in [3], Alzaid et al.'s scheme suffers from a new kind of attack called the sandwich attack. Two nodes are captured at times t_i and t_j ($t_i < t_j$), respectively, reveals all the group keys used between times t_i and t_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 sensor node at time t_i which then leads into 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 t_j, he can compute all the preimages 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 t_i to t_j by the computation: $K_G^k := h^k(s_0) \oplus h^{-k}(t_0)$, where $t_i \leq t_k \leq t_j$.

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 protocol is a modified version of Protocol 1 to accommodate this idea.

Protocol 3: The modified version of the group key update protocol in Alzaid et al.'s scheme [3]

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, *reset the value of* $h^{-i}(t_0)$ to $t_0^{'}$, and update the value of the group key ($K_G^i := h^i(s_0) \oplus h^{-i}(t_0)$).

The protocol messages of Protocol 3 are exactly the same as those of Protocol 1 except for the addition of a new data $t_0^{'}$. 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 $t_0^{'}$. In other words, M reestablishes the reverse hash chain with $t^{'}$ as a starting point.

It should be noted that, after the execution of Protocol 3, $h^{-i}(t_0)$ 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 $t_0^{'}$, i.e., $h^{-i}(t_0) := t_0^{'}$. 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 originated from the network manager. Note that $t_0^{'}$ is delivered to the sensor node encrypted under the pairwise key K_{MN}, not under the group key. And then, the new group key, which is computed by using the new reverse hash chain, is hashed and then returned to the network manager. Thus, the network manager can be certain that $t_0^{'}$ has successfully installed into the sensor node.

Interestingly, the modified protocol equipped with the countermeasure comes with a nice feature: reestablishing the reverse hash chain. With this feature, the sensor nodes do not have to be recollected to refill the reverse hash chain. Now, the network manager can initiate Protocol 3 any time to restart the reverse hash chain, hence arbitrarily limiting the time span within which sandwich attacks succeed.

In fact, M can play two strategies in order to accomplish the reinitialization of the reverse hash chain. On one hand, M can replace Protocol 1 completely with Protocol 3. The only drawback with this strategy is that the self-synchronization feature, as mentioned in the description of Protocol 1, cannot be maintained anymore. Therefore, M must rerun Protocol 3 until he receives the second message of the protocol from N to ensure that the reverse hash chain has been reestablished. In return, however, we get a key management entirely free from sandwich attacks.

On the other hand, M can switch between Protocol 1 and Protocol 3 whenever it is needed. For example, the network manager can use only Protocol 1 to renew the group key several times based on the same reverse hash chain. When there is a suspicion that the sandwich attack may occur, M can switch and run Protocol 3 in order to limit the usefulness of the disclosed components of the previous hash chain. After that, M can switch back to Protocol 1. The choice between these two strategies depends on how much concern the network designer has with the sandwich attack.

4 Performance Analysis

Due to the resource constraints in WSNs [2], new metrics such as energy efficiency have been introduced in order to validate the performance of new proposals. In this section, we analyze the performance of our

Table 2: Memory Overhead Comparison

Stored information per sensor	Nilsson et al. [15]		Alzaid et al. [3]		Our proposal	
	Qty	Size (bits)	Qty	Size (bits)	Qty	Size (bits)
Pairwise key shared with M (K_{MN})	2	256	1	256	1	256
Key used for random number generation	1	128	-	-	-	-
$M^{'}$s public key	1	256	-	-	-	-
Group key (K_G)	1	128	1	128	1	128
Secret data	-	-	2	128	2	128
Indexes	-	-	2	16	2	16
Hashed value of the old pairwise key	-	-	1	128	1	128

proposal, Alzaid et al.'s scheme [3], and the similar scheme proposed by Nilsson et al. [15]. Our performance analysis covers memory overhead, communication cost, and computation cost for these schemes. To the best of our knowledge, the performance analysis for these schemes does not exist in the current literature.

4.1 Memory Overhead

In this section, we discuss the amount of memory required by our proposal. Interestingly, it does not need more memory than what is required by Alzaid et al.'s scheme. This is because each sensor node replaces t_0 with $t_0^{'}$ when the network manager reestablishes the reverse hash chain as discussed in Section 3. There is no need to keep copies of both t_0 and $t_0^{'}$ at the same time, only one of them is needed.

Each sensor node, prior to the deployment phase, stores four pieces of information: the secret data ($h^i(s_0)$ and $h^{-i}(t_0)$), two indexes: one for the group key update phase (i) and another one (j) to handle the delivery failure problems. The delivery failure problem is not discussed in this paper because it was not affected by the sandwich attacks. However, it is an important part of the proposed protocol.

The sensor node then needs to keep a copy of the recent pairwise key shared with M (which is K_{MN}), the group key (which is K_G), and a hashed copy of the old pairwise key (which is $h^j(K_{MN})$. The reason for keeping a hashed copy of the old pairwise key is to use it when M runs the delivery failure protocol as described in Alzaid et al.'s scheme. In other words, a sensor node needs to store two symmetric keys (K_{MN}, K_G), two secret values ($h^i(s_0)$, $h^{-i}(t_0)$), two indexes (i, j), and a hashed copy of the previous pairwise key.

Consequently, each sensor node, in our proposal and Alzaid et al.'s scheme, needs to store approximately 100 bytes in order to achieve 128 bit security. This memory overhead occupies approximately 0.078% of the total program flash memory at the most popular sensor end device mica2 [6]. The 100 bytes include two keys (256 bit and 128 bit long each), two 128 bit secret data, two 16 bit indexes, and one 128 bit hashed value of the previous pairwise key (see Table II).

On the other hand, Nilsson et al.'s scheme occupies approximately 128 bytes which is equivalent to 0.1% of the total program memory in order to achieve the same level of security. This memory overhead includes two 256 bit pairwise keys between M and N (one is the current pairwise key and the other is a copy of the previous key to handle the key delivery failure), one pre-installed 128 bit secret key that is used to generate the random number, one 256 bit public key for M, and one 128 bit group key (see Table II).

4.2 Communication Overhead

The communication between sensor nodes is considered the biggest factor that destroys the sensor's battery since it consumes most of the available power. It consumes much more than sensing and computation activities. Hill et al. concluded that each bit transmitted in WSNs consumes about as much power as executing 800-1000 instructions [11]. The mica2 data sheet declared that the energy consumption of communication, which is the focus of this section, is unequal for sending and receiving [6]. The energy consumption of transmitting with maximum power is more than double the energy consumption of receiving activities.

On one hand, the energy consumption for transmitting m bits over a distance r, according to [1, 10], can be calculated as follows:

$$E_{tx}(m,r)(m,r) \;=\; mE_c \;+\; mer^s \;,$$

$$\text{where} \quad \text{e}=\begin{cases} e_1 s = 2, & for\ r < r_{cr} \\ e_2 s = 4, & for\ r > r_{cr} \end{cases} \quad (1)$$

Here E_c represents the minimum energy required to operate the radio circuit, e denotes the unit energy required for the transmitter amplifier, and r_{cr} is the crossover distance. The typical values for E_c, e_1, and r_{cr} are $50\,nJ/bits$ for a 1 Mbps transceiver, $10\,pJ/bit\,m^2$, and $86.2\,m$, respectively.

On the other hand, the energy consumption that is resulted from the receiving activities can also be calculated as follows:

$$E_{rx}(m,r) = mE_c \quad (2)$$

Assuming that $r = 50 < r_{cr}$, Table III lists the number of bits that is required to be transmitted in order to accomplish the renewal of the pairwise and group keys.

In our proposal, the network manager may need to reset the reverse hash chain in order to defeat the sandwich attack as in Protocol 3. The repetition of this process depends on the time span, which is defined by M, within which the adversary is allowed to succeed in launching the sandwich attack. M may reset the reverse hash chain every time when he updates the group key, or he may reset it after l group key renewals. It depends totally on the protocol specification.

In comparison with the group key update protocol (Protocol 1) in Alzaid et al.'s scheme, the reestablishment of the reverse hash chain (Protocol 3) requires the sensor node N to receive 128 more bits in the first message of the protocol (see Table III). The inclusion of the extra 128 bits, which is $t_0^{'}$, with $h^{-i}(t_0)$ in the first message is important to convince the sensor node that the reestablishment of the reverse hash chain is originated by M as discussed in Section 3. After the success of running Protocol 3, M can run Protocol 2 in order to update the group and pairwise keys at the same time.

The transmission of those extra bits leads to more energy consumption. Table III shows that our proposal consumes $6.4\,J$ more energy than Alzaid et al.'s scheme in order to update the group key. M, in our proposal, may run Protocol 1 if there is no need to reestablish the reverse hash chain. This means that the increase in the energy consumption is not continuous, and it exists only when there is a need for the reestablishment. If there is no need to reset the reverse hash chain, the transmission energy consumption for our proposal is the same as Alzaid et al.'s scheme.

The network manager M, in our proposal and in Alzaid et al.'s scheme, initiates the pairwise key rekeying mechanism while the sensor node itself initiates the mechanism in Nilsson et al.'s scheme. Our proposal and Alzaid et al.'s scheme require M and N to swap the Diffie-Hellman components g^{r_M} and g^{r_N}. This increases the length of the information received by a sensor node by 34 bytes compared to Nilsson et al.'s scheme. Although this increase affects the energy consumption, it is a must to solve the security weaknesses in Nilsson et al.'s scheme. We refer interested readers in these weaknesses to [3]. Importantly, this increase in the number of transmitted bits affects the energy consumption of receiving activities (E_{rx}), but not the energy

Table 3: Number of Bits Transmitted/Received by A Sensor

Protocol	Step	Nilsson et al. [15]		Alzaid et al. [3]		Our proposal	
		# of bits	Consumed energy (μJ)	# of bits	Consumed energy (μJ)	# of bits	Consumed energy (μJ)
Pairwise Key	1. $M \rightarrow N$	-	-	272	13.6	272	13.6
	2. $M \leftarrow N$	256	19.2	256	19.2	256	19.2
	Total	**256**	**19.2**	**528**	**32.8**	**528**	**32.8**
Group Key	1. $M \rightarrow N$	256	12.8	144	7.2	272	13.6
	2. $M \leftarrow N$	128	9.6	128	9.6	128	9.6
	Total	**384**	**22.4**	**272**	**16.8**	**400**	**23.2**

consumption of transmitting activities (E_{tx}). Moreover, the pairwise key rekeying mechanism (Protocol 2) is able to update the pairwise and group keys at the same time, especially if there is no indication for any node compromise attack. Table III shows that the communication energy consumption ($E_{tx} + E_{tx}$) that results from updating these two keys is $32.8\,J$ in our proposal and Alzaid et al.'s scheme while it is $41.6\,J$ in Nilsson et al.'s scheme.

As already mentioned, M runs Protocol 2 in order to update the pairwise and group keys in the same time. However, M sometimes may need to remove specific nodes from a particular group, especially when they behave maliciously. In this case, M can run Protocol 1, in Alzaid et al.'s scheme, or Protocol 3, in our proposal. Table III shows that Protocol 3 (Protocol 1) requires sensor nodes in our proposal (in Alzaid et al.'s scheme) to receive 2 bytes more (14 bytes less) than Nilsson et al.'s scheme in the first message of the group key update protocol. However, Protocols 3 (Protocol 1) in our proposal (Alzaid et al.'s scheme) sends the same number of bits as Nilsson et al.'s scheme in the second message of the group key update protocol.

4.3 Computation Cost

We assess, in this section, the energy consumption that results from applying cryptographic operations in our proposal, and then compare this consumption with those of Alzaid et al.'s and Nilsson et al.'s schemes as in Table IV.

For concreteness, we assume that $RC5$ is used for symmetric encryption/decryption activities, $SHA-1$ is used for hash operations, and $ECDSA$ is used for public key encryption. We estimate the cost of the cryptographic operations based on the results from some analysis studies presented in [5, 8, 17, 18].

To update the pairwise key, our proposal consumes the same as Alzaid et al.'s scheme since the pairwise key update protocol in both schemes are the same. However, our proposal and Alzaid et al.'s schemes consume $274\,\mu J$ more energy in comparison with Nilsson et al.'s scheme.

On the other hand, each sensor node in our proposal consumes $26\,\mu J$ more energy to update the group key in comparison with Alzaid et al.'s scheme. In Protocol 3, M encrypts the new seed of the reverse hash chain with the next preimage of the current reverse hash chain. This encrypted message is longer by 128 bits than the first message of Protocol 1 in Alzaid et al.'s scheme. This means that N in our proposal, upon receiving this message, needs to decrypt it with a cost of $26\,\mu J$ more energy than in Alzaid et al.'s scheme. In other words, this extra energy consumption in Protocol 2 comes as a result of decrypting longer messages. It is worth mentioning that the $26\,\mu J$ increase in the computation energy consumption, in comparison with Alzaid et al.'s scheme, is not continuous. It exists only when there is a need to reset the reverse hash chain. The repetition of this process depends on the time span, which is defined by M, within which the adversary is allowed to succeed in launching the sandwich attack. M may reset the reverse hash chain every time when he updates the group key, or he may reset it after l group key renewals. It depends totally on the protocol

Table 4: Computation Cost

Protocol	Step	Consumed Energy (μJ)		
		Nilsson et al. [15]	Alzaid et al. [3]	Our proposal
Pairwise Key	1. $M \rightarrow N$	-	304	304
	2. Compute the new key	154	52000	52000
	3. $M \leftarrow N$	52154	278	278
	Total	**52308**	**52582**	**52582**
Group Key	1. $M \rightarrow N$	150	278	304
	2. Compute the new key	-	154	154
	2. $M \leftarrow N$	154	154	154
	Total	**304**	**586**	**612**

specification.

Interestingly, our proposal and Alzaid et al.'s can update the pairwise and group keys at the same time by running Protocol 2, especially if there is no need to eliminate some group members from the group. This can be done by a sensor node with computation cost of $52582\,\mu J$. However, Nilsson et al.'s scheme needs to run both protocols, pairwise key update protocol and group key update protocol, to update pairwise and group keys with total computation cost of $52612\,\mu J$.

5 Conclusion

Alzaid et al.'s key management scheme has nice features: forward and backward secrecy or future/past key secrecy as called in [3]. 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 the sandwich attack which is a special kind of node capture attack. Another potential issue with the scheme was how to refill the reverse hash chain.

We take advantage of the message authentication feature of the first message of Protocol 1 of Alzaid et al.'s scheme to deliver a commitment ($t_0^{'}$) from the network manager to the sensor nodes (as shown in Protocol 3). This commitment data is now used as the starting value for a "new" reverse hash chain. Our proposal (Protocol 3) does not necessarily replace the original group key update protocol (Protocol 1); Protocol 3 can be used only when the network manager needs to reinitialize the reverse hash chain and mitigate the sandwich attacks. Thus, the sandwich attack can only be attempted with a very limited time interval which corresponds to the lifetime of a particular reverse hash chain. In short, our countermeasure against sandwich attacks makes Alzaid et al.'s scheme more resilient against node capture attacks and supports reinitialization of the reverse hash chain at the same time.

In an application environment where the prevention of the sandwich attack, not just a mitigation of the attack, is strictly needed, our proposal provides the solution. It replaces Alzaid et al.'s original protocol (Protocol 1) with Protocol 3 in our proposal. The only drawback of this solution is the self-synchronization property, as mentioned in the description of Protocol 1 in Section II, is not maintained anymore.

Our performance analysis shows that the sensor node in our proposal consumes approximately $52614.8\,\mu J$ and $635.2\,\mu J$ in order to update the pairwise key and the group key, respectively. This energy consumption includes the communication cost and the computation cost as listed in Tables III and IV. Our proposal's energy consumption for the pairwise key update is the same as Alzaid et al.'s scheme, but it is $287.6\,\mu J$ more than Nilsson et al.'s scheme. This difference is due to the security enhancements that are required

to overcome the weaknesses in Nilsson et al.'s scheme as discussed in Section II. To update the group key, our proposal consumes 32.4 μJ and 308.8 μJ more energy than Alzaid et al.'s and Nilsson et al.'s schemes, respectively. This additional cost is for defeating the sandwich attack and overcoming the weaknesses of Nilsson et al.'s scheme.

Acknowledgment

The research of the first author was supported by King Abdulaziz City for Science and Technology[1] in Saudi Arabia.

References

[1] A.A. Ahmed, H. Shi, and Y. Shang. A survey on network protocols for wireless sensor networks. In *Proceedings of the international conference on Information Technology: Research and Education, ITRE'03*, pages 301–305, August 11-13, 2003.

[2] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, March, 2002.

[3] Hani Alzaid, DongGook Park, Juan González Nieto, Colin Boyd, and Ernest Foo. A forward and backward secure key management in wireless sensor networks for PCS/SCADA. In *proceedings of the* 1^{st} *International ICST Conference on Sensor Systems and Software, Pisa, Italy*, September 7-8, 2009, in press.

[4] C. L. Beaver, D.R. Gallup, W. D. NeuMann, and M.D. Torgerson. Key management for SCADA. Technical report, Sandia National Laboratories, March, 2002. . [Online]. Available: `www.sandia.gov/scada/documents/013252.pdf` [Accessed: October 10, 2009].

[5] Chih-Chun Chang, Sead Muftic, and David J. Nagel. Measurement of energy costs of security in wireless sensor nodes. In *Proceedings of the* 16^{th} *International Conference on Computer Communications and Networks, IEEE ICCCN'07*, pages 95–102, August 13-16, 2007.

[6] Crossbow Technology Inc. Mica2 datasheet, 2006. . [Online]. Available: `http://www.xbow.com/Products/productdetails.aspx?sid=174` [Accessed: October 10, 2009.

[7] Robert Dawson, Colin Boyd, Ed Dawson, and Juan Manuel González Nieto. SKMA: a key management architecture for SCADA systems. In Rajkumar Buyya, Tianchi Ma, Reihaneh Safavi-Naini, Chris Steketee, and Willy Susilo, editors, *Proceedings of the* 4^{th} *Australasian Symposium on Grid Computing and e-Research (AusGrid'06) and the* 4^{th} *Australasian Information Security Workshop (Network Security) (AISW'06)*, volume 54 of *CRPIT*, pages 183–192. Australian Computer Society, January, 2006.

[8] Giacomo de Meulenaer, François Gosset, François-Xavier Standaert, and Olivier Pereira. On the energy cost of communication and cryptography in wireless sensor networks. In *Proceedings of the* 4^{th} *IEEE International Conference on Wireless & Mobile Computing, Networking & Communication, WIMOB'08*, pages 580–585. IEEE Computer Society, October 12-14, 2008.

[9] Carl Hartung, James Balasalle, and Richard Han. Node compromise in sensor networks: The need for secure systems. Technical report, University of Colorado at Boulder, January, 2005. . [Online]. Available: http://www.cs.colorado.edu/department/publications/reports/docs/CU-CS-990-05.pdf [Accessed: October 10, 2009].

[10] W.B. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1(4):660–670, October 2002.

[1] http://www.kacst.edu.sa

[11] Jason L. Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David E. Culler, and Kristofer S. J. Pister. System architecture directions for networked sensors. In *Proceedings of the* 9^{th} *International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS'00, Cambridge, MA, USA*, pages 93–104, November 12-15, 2000.

[12] Marek Klonowski, Miroslaw Kutylowski, Michal Ren, and Katarzyna Rybarczyk. Forward-secure key evolution in wireless sensor networks. In Feng Bao, San Ling, Tatsuaki Okamoto, Huaxiong Wang, and Chaoping Xing, editors, *Proceedings of the* 6^{th} *International Conference on Cryptology and Network Security,CANS'07*, volume 4856 of *Lecture Notes in Computer Science*, pages 102–120. Springer, December 8-10, 2007.

[13] Sjouke Mauw, Ivo van Vessem, and Bert Bos. Forward secure communication in wireless sensor networks. In John A. Clark, Richard F. Paige, Fiona Polack, and Phillip J. Brooke, editors, *Proceedings of the* 3^{rd} *International Conference on Security in Pervasive Computing, SPC'06*, volume 3934 of *Lecture Notes in Computer Science*, pages 32–42. Springer, April 18-21, 2006.

[14] Robert McClanahan. SCADA and IP: is network convergence really here? *Industry Applications Magazine, IEEE*, 9(2):29–36, Mar/Apr 2003.

[15] Dennis K. Nilsson, Tanya Roosta, Ulf Lindqvist, and Alfonso Valdes. Key management and secure software updates in wireless process control environments. In Virgil D. Gligor, Jean-Pierre Hubaux, and Radha Poovendran, editors, *Proceedings of the* 1^{st} *ACM Conference on Wireless Network Security, WISEC'08*, pages 100–108. ACM, March 31 - April 02, 2008.

[16] L. Pietre-Cambacedes and P. Sitbon. Cryptographic key management for SCADA systems-issues and perspectives. In *Proceedings of the* 2^{nd} *international conference on Information Security and Assurance, ISA'08*, pages 156–161. IEEE Computer Society, April, 2008.

[17] Ramnath Venugopalan, Prasanth Ganesan, Pushkin Peddabachagari, Alexander Dean, Frank Mueller, and Mihail Sichitiu. Encryption overhead in embedded systems and sensor network nodes: modeling and analysis. In *Proceedings of the international conference on Compilers, architecture and synthesis for embedded systems, CASES'03*, pages 188–197, October 30 - November, 12003.

[18] Arvinderpal Wander, Nils Gura, Hans Eberle, Vipul Gupta, and Sheueling Chang Shantz. Energy analysis of public key cryptography for wireless sensor networks. In *Proceedings of the* 3^{rd} *IEEE International Conference on Pervasive Computing and Communications, PerCom'05*, pages 324–328. IEEE Computer Society, March, 2005.