Sensor Authentication in Dynamic Wireless Sensor Network Environments

Kyusuk Han
KAIST, Korea

Taeshik Shon
Ajou University

Abstract

While applications of Wireless Sensor Network are diversified, several new issues such as mobility of sensor node are raised and bring security issues such as re-authentication and tracing the node movement. In the dynamic sensor network, mobile sensor nodes will continuously move around and frequently reconnect to other sensor nodes. While many security protocols to such networks occur significantly large overheads because their design only considered the static networks. There are several studies on such dynamic environments. In this paper, we show our design for the efficient node authentication and key exchange that reduces the overhead in node re-authentication and also provides untraceablity of mobile nodes. We introduce protocols that are symmetric key crypto system based and public key crypto system based. We also introduce the application scenario of the protocol that is integrated to other networks.

1. Introduction

Wireless Sensor Network (WSN) is the network that consists of lightweight devices with short-ranged wireless communication and battery-powered. The devices have the sensor that gathers the environmental information and etc. After sensing this information, the devices send the information to the networks. We define such devices as sensor node, and the core parts of the network as sinks and the base station.

Rapid development of WSNs brought themselves to be deployed to various areas such as RF4CE [19]. However, many security studies on WSNs are rather inefficient to be deployed to such environments.

While there are many trials for providing efficient security functions for WSN such as [4, 5, 13, 14], as one of the fundamental security issues, there are various researches on key management in WSN such as key-pre distribution, pairwise key agreement, group key based key agreement, and hierarchical key management schemes. There were also trials of PKI deployment for WSN [17]. In order to reduce the communication overhead from the key establishment, Huang *et al.* [10] proposed public key infrastructure (PKI) based model applying Elliptic Curve Cryptography [15]. However, applying PKI requires larger computational power, although it enables the simplified key agreement procedure.

Since it is obvious that the wireless sensor network will be widely deployed by combining network of static sensor network and mobile entities such as [2, 6], handling a large overhead from frequent node re-authentication requests due to the continuous node movements and the threats of tracing the node movement will be important security issues.

In order to solve such problems, we proposed efficient authentication models that significantly reduce overhead for re-authenticating sensor nodes [7, 8].

In this paper, we show our model that provide high efficiency in such environments and shows several protocols that we previous introduced in [7, 8].

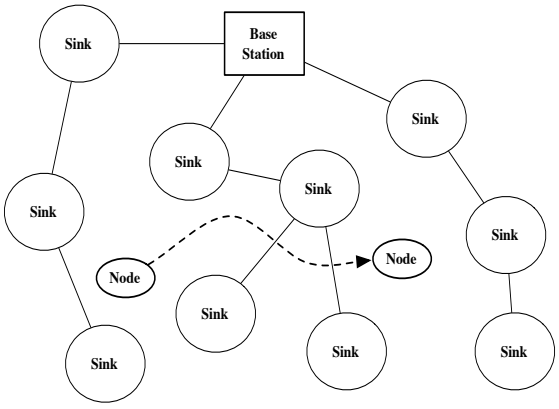

Figure 1 A dynamic mobile node continuously moves in the sensor networks that static sinks established. The unbroken line denotes the static connection between sinks and the base station. The dotted line denotes the movement of the mobile node.

2. Our Sensor Authentication Model

2.1. Dynamic WSN Environment

In this section, we claim the security issues on the node mobility in WSN and problems of previous authentication and key agreement models. In paper

[kyusuk1, kyusuk2, kyusuk3], we defined a sensor network model with moving nodes as in Figure 1. We also defined a static sensor node as Sink, a mobile node as Node, and the base station that is the core network. The node has linear movements in the network. The base station and sinks are static as same as Ibriq and Mahgoub's model [11]. Sinks act as the gateway that link nodes to the base station, and the base station is a kind of headquarter that manages entire networks. When a node initially joins the network, the node connects to a sink in the network and is authenticated by the sink with help of the base station. After that the node moves and reconnects to other sink. In the model, the sink that re-authenticates the node is the neighbor sink of the sink that previously authenticated the node. The re-authentication processes frequently happen due to the node continuously moves in the network.

In practical scenarios, re-authentication happens when a node lost connection to the sink or moved and connected to other sink. For the former case, the node can be easily re-authenticated to the same sink when the connection becomes available again. For the latter case, the node request the re-authentication to other sink that is the near to the previously attached sink.

For such environments several security Issues raise as followings.

2.1.1. Frequent Re-authentication. Since the sensor have low powered battery and low-end processor with short-range wireless communication, the reducing communication and computational overheads is important to increasing the lifetime of the sensor. However, the mobile sensor node may occur the large overhead for the security computation due to the frequent requests of node re-authentication. When a node connects to a sink, the sink has to authenticate the node. When the node connects to other sink after the movement, the new sink has to authenticate the node again. If the node has continuous movement, the authentication process will also occur repeatedly. It is obvious that the frequent re-authentication processes are the significant factors that drain the resources in battery-based sensor nodes.

However, the current authentication and key distribution protocols are insufficient to be applied in such environment with lack of consideration of the node mobility. Using the current protocols such as [11, 1], the communication overhead for re-authentication is as same as previous authentication.

Such overhead will be the problem in the environment that the frequent movements of the large number of nodes are happened. Thus, the less computational and communication overheads in re-authentication are very urgent requirement for the node mobility support in the WSN.

2.1.2. Tracing Node Movements. Considering the mobility of sensor nodes, the tracking of the node movement is one of possible attacks. When the mobile nodes are deployed in battlefields, the tracking by enemies is significant threats for the networks. Thus, the authentication and key agreement protocols should not reveal the node movement.

2.2 Security Requirements

We defined the security requirements as follows. We assume that when the node N communicates with a sink S_2 after disconnection to the sink S_1, S_1 cannot receive any message between N and S_2. S_2 is one of neighbor sinks of S_1.

- **Re-authentication** An authenticated node N and S_2 should be able to identify each other with less communication and computational overhead than initial authentication.
- **Untraceablity** In re-authentication of N, S_2 only knows that N was previously connected to S_1, and never knows the direction of N.
- **Confidentiality** When N and S_1 are operating initial authentication, nobody can know the communication packet between N and S_1, between S_1 and BS. For re-authentication between N and S_2, nobody except S_1 can know the communication information, while S_1 out of communication range.
- **Message Integrity** Any malicious adversaries should not be able to forge the communication packet.
- **Key Freshness** N and S should be able to verify that the key is generated during the current session.
- **Node/Sink Resiliency** Even N, S_1 or S_2 are compromised by a malicious adversary, they should not be able to affect to the entire network.

'Confidentiality', 'message Integrity', and 'key freshness' are important requirements against the replay attack or man-in-the-middle attack. 'Node/Sink resiliency' is practical threat that the sensor nodes are generally deployed in the environment out of administration.

3. Efficient Sensor Authentication Protocol for Dynamic Environments

3.1. Overview of Our Model

We briefly describe our proposed protocol. Assume that there are a base station BS, a sink S_1, a neighbor sink S_2, and a mobile node N in the

network. S_1 periodically broadcasts HELLO in Phase 0. When S_2 receives HELLO, S_2 initiates the neighbor relationship if S_1 is a newly discovered sink. After the pairwise key between S_1 and S_2 has been exchanged in Phase 1, S_1 and S_2 exchange the authentication key that is used to verify the authenticated user in Phase 2. Phase 1 and Phase 2 are bootstrapping phase for the node re-authentication.

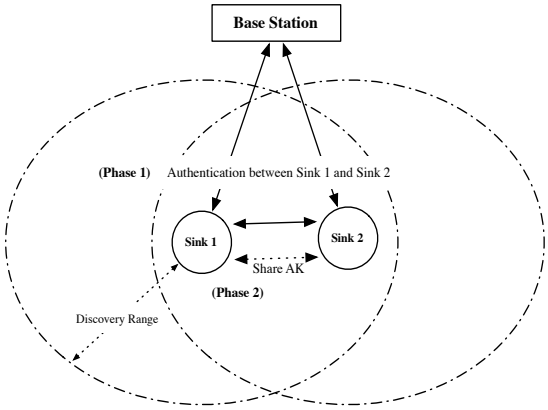

Figure 2 Sink 1 (S_1) mutually authenticate Sink 2 (Phase 1), and share the authentication key (Phase 2).

When N that is not authenticated by any sink joins the network, N receives HELLO of S_1 and initiates the initial node authentication with S_1 in Phase 3. Later, N moves and reconnects to S_2. Then N initiates the node re-authentication in Phase 4. Figure 2, 3 shows each phase of our model.

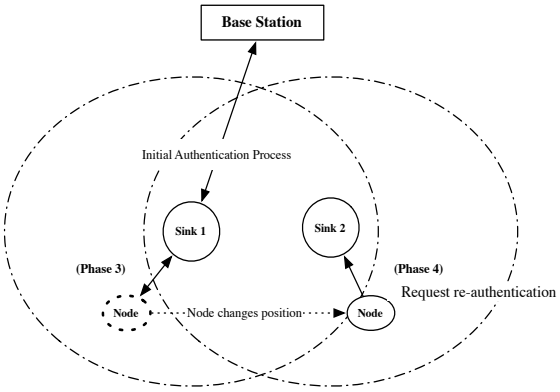

Figure 3 Node is initially authenticated by Sink 1 (Phase 3), and requests re-authentication to Sink 2.

Thus our model consists of following five phases.

- **Phase 0** The common neighbor discovery
- **Phase 1** Setting up neighbor sink relationship
- **Phase 2** Neighbor group authentication key share
- **Phase 3** Initial node authentication
- **Phase 4** Node re-authentication

3.1.1. Authentication Ticket. We define 'Authentication Ticket' that is used for the node re-authentication. When a node requests authentication to a sink, the sink generates the authentication ticket and sends it to the node. The authentication ticket is verified by the authentication key shared to neighbor sinks. Using the authentication ticket, the node movement is untraceable. Verification of the authentication ticket is available to neighbor sinks of the sink that issued the ticket. We adopt the idea of 'cluster key' in [LEAP+] that shared to neighbor sinks. The main difference is that the role of cluster key is very limited to verifying the authentication ticket. Thus, we rename the key as 'authentication key' due to the limited use in the protocol.

3.1.2. Neighbor Sink List. Assume that static sink nodes are distributed as shown in Figure 4 (a). In this case, the node authenticated by S_3 can be re-authenticated from the neighboring sinks S_1 or S_4, wherever it moves. However, the sinks may not be well distributed in the real environments.
In Figure 4 (b), a node that is authenticated by S_1 cannot directly be re-authenticated by S_5, since S_5 is not a neighbor sink of S_1. However, we also see that both S_1 and S_5 have the common neighbor sink S_2 that may link S_1 and S_5 for the re-authentication of N.

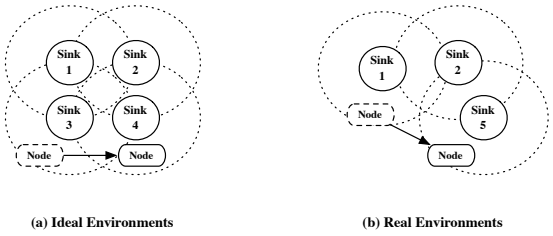

Figure 4: Static sensor node distribution in (a) Ideal environments (b) Real Environments

Thus, we defined the NSL that stores the neighboring sink's information such as their ID, shared secret key, and public key. We assume that each sink has its own NSL. Let the NSL stored in S_1 be $NSL_{S_1} = t \| sign_{S_1}(h(t))$, where $t = S_1 \| S_2 \| .... \| S_k$ and $sign_{S_1}$ is the signature of S_1 as in Table 1. Figure 8 shows that S_1 and S_5 have a common sink S_2 that can link the two sinks. The signature scheme can be flexibly chosen. When TinyECC [15] is used, the Elliptic Curve Digital Signature Algorithm (ECDSA) can be properly applied. Whenever a new neighboring sink is found, the sink updates its own NSL.

3.2. Symmetric Key based Protocol

3.2.1. Pre-Phase 0: Neighbor Discovery. A sink S_1 periodically generates a random nonce R_1 . S_1 also generates $u_0 = E_{K_{S_1}}\{R_0 \| TS_0\}$ and

$v_0 = MAC_{K_{S_1}}(S_1 \| \text{HELLO} \| u_0)$, where TS_0 is time stamp. u_0 and v_0 are included in the HELLO message. Then S_1 broadcasts u_0 and v_0 as follows:

$$S_1 \rightarrow Broadcast : S_1 \| HELLO \| u_0 \| v_0$$

Phase 0 is the periodical common procedure. When a sink receives HELLO, the sink initiates Phase 1 or Phase 2. When a node receives HELLO, the node initiates Phase 3 or Phase 4.

3.2.2. Pre-Phase 1: Setting up Neighbor Sink Relationship. Assume another sink S_2 receives HELLO message. S_2 checks the sender of HELLO whether S_1 is known or not. If S_2 already knows S_1, S_2 discards the message. Otherwise, S_2 requests the setting up the neighbor relationship as follows:

1. S_2 randomly selects R_1 and generates $u_1 = E_{K_{S_2}}\{R_1 \| u_0\}$, $v_1 = MAC_{IK_{S_2}}(S_2 \| \text{BS} \| S_1 \| u_1 \| v_0)$.

$$S_2 \rightarrow \text{BS} : S_2 \| \text{BS} \| S_1 \| u_1 \| v_1 \| v_0$$

2. After verifying v_1, BS decrypts u_1 and retrieves R_1 and v_0. Then, BS verifies v_0 and decrypts v_0. Finally, BS retrieves R_0 and TS_0. BS generates and sends u_4, v_4, and v_3 to S_2 where, $u_3 = E_{K_{S_1}}\{R_1\}$, $v_3 = MAC_{IK_{S1}}(\text{BS} \| S_1 \| u_3)$, $u_4 = E_{K_2}\{R_1 \| u_3\}$ and $v_4 = MAC_{IK_2}(\text{BS} \| S_2 \| R_1 \| u_4 \| v_3)$.

$$BS \rightarrow S_2 : BS \| S_2 \| S_1 \| u_4 \| v_4 \| v_3$$

3. After verifying v_4, S_2 decrypts u_4, and retrieves R_1 and u_3. S_2 generates $K_{S_1S_2} = KDF(0 \| R_0 \| R_1)$ and $IK_{S_1S_2} = KDF(1 \| R_0 \| R_1)$ with R_0 and R_1. $K_{S_1S_2}$ is encryption key and $IK_{S_1S_2}$ is integrity key between S_1 and S_2. Then S_2 generates $v_5 = MAC_{IK_{S_1S_2}}(S_2 \| S_1 \| R_0 \| R_1)$ and sends u_3, v_3, and v_5 to S_1.

$$S_2 \rightarrow S_1 : S_2 \| S_1 \| u_3 \| v_3 \| v_5$$

4. After verifying v_3, S_1 decrypts u_3 and retrieves R_1. S_1 also generates $K_{S_1S_2}$ and $IK_{S_1S_2}$. Then S_1 verifies v_5. S_1 generates $v_6 = MAC_{IK_{S_1S_2}}(S_1 \| S_2 \| ACK \| R_0 \| R_1)$ and sends v_6 with ACK to S_2.

$$S_1 \rightarrow S_2 : S_1 \| S_2 \| ACK \| v_6$$

5. S_2 verifies v_6 and shares pairwise keys $K_{S_1S_2}$ and $IK_{S_1S_2}$.

3.2.3. Phase 2: Neighbor Group Authentication Key Share. Phase 2 can be operated solely or after Phase 1 is completed. In Phase 2, S_1 initiates following procedures.

1. S_1 randomly selects nonce $ASEED_{S_1}$ and R_1. Then S_1 generates $u_1 = E_{K_{S_1S_2}}\{ASEED_{S_1} \| R_1\}$ and $v_1 = MAC_{IK_{S_1S_2}}(S_1 \| S_2 \| u_1)$.

$$S_1 \rightarrow S_2 : S_1 \| S_2 \| u_1 \| v_1$$

2. After verifying v_1, S_2 decrypts u_1, and retrieves $ASEED_{S_1}$ and R_1. Then S_2 generates $AK_{S_1} = KDF(0 \| ASEED_{S_1})$ and $AIK_{S_1} = KDF(1 \| ASEED_{S_1})$. S_2 also generates $v_2 = MAC_{AIK_{S_1}}(S_2 \| S_1 \| \text{ACK} \| R_1)$ using AIK_{S_1}.

$$S_2 \rightarrow S_1 : S_2 \| S_1 \| \text{ACK} \| v_2$$

3. S_1 verifies v_2.

After the Phase 2 is completed, sinks share their neighbor sink's authentication keys.

3.2.4. Phase 3: Initial Node Authentication. When N receives HELLO that S_1 broadcasts in Phase 0 and is not yet authenticated by any sink, N proceeds followings.

1. Node N randomly selects R_1 and generates $u_1 = E_{K_N}\{R_1 \| u_0 \| v_0\}$ and $v_1 = MAC_{IK_N}(N_1 \| S_1 \| u_1)$.

$$N \rightarrow S_1 : N \| S_1 \| u_1 \| v_1$$

2. S_1 generates $v_2 = MAC_{IK_{S_1}}(S_1 \| \text{BS} \| N \| u_1 \| v_1)$.

$$S_1 \rightarrow \text{BS}: S_1 \| \text{BS} \| N \| u_1 \| v_1 \| v_2$$

3. After verifying v_2 and v_1, BS decrypts u_1, and retrieves R_0, u_0 and v_0. After verifying v_0, BS decrypts u_0, and retrieves R_0 and TS. BS checks the validity of TS and generates $u_3 = E_{K_N}\{R_0\}$, $v_3 = MAC_{IK_N}(BS \| N \| S_1 \| u_3)$, $u_4 = E_{K_{S_1}}\{R_1 \| u_3 \| v_3\}$ and $v_4 = MAC_{IK_{S_1}}(BS \| S_1 \| N \| R_0 \| u_4)$.

$$BS \rightarrow S_1 : BS \| S_1 \| N \| u_4 \| v_4$$

4. After verifying v_4, S_1 decrypts u_4, and retrieves R_1, u_3 and v_3. Then S_1 generates $NK_N = KDF(R_0 \| R_1)$. S_1 generates $t = E_{AK_{S_1}}\{TS \| R_1 \| NK_N\}$ and $w = MAC_{AIK_{S_1}}(N \| t)$. Next, S_1 also generates $u_5 = E_{NK_N}\{TS \| t \| w\}$ and $v_5 = MAC_{NIK_N}(S_1 \| N \| R_0 \| u_5)$.

$$S_1 \rightarrow N : S_1 \| N \| u_3 \| v_3 \| u_5 \| v_5$$

5. After verifying v_3, N decrypts u_3 and retrieves R_0. Then N also generates NK_N and verifies v_5. N decrypts u_5 and retrieves TS, t and w. N generates $v_6 = MAC_{NK_N}(N \| S_1 \| ACK \| R_0 \| R_1)$.

$$N \rightarrow S_1 : N \| S_1 \| \text{ACK} \| v_6$$

6. S_1 verifies v_6.

3.2.5. Phase 4: Node Re-authentication. When N receives HELLO that S_2 broadcasts in Phase 0 and is previously authenticated by a sink, N proceeds followings.

1. N generates $v_1 = MAC_{NIK_N}(N \| S_2 \| t \| w \| v_0)$.

$$N \rightarrow S_2 : N \| S_2 \| t \| w \| v_1$$

2. S_2 verifies w and decrypts t. S_2 retrieves R_1, NK_N and TS. Using NK_N, S_2 verifies v_1. Then S_2 generates $NK' = KDF(R_1 \| R_0)$, also generates $t' = E_{AK_{S_2}}\{R_1 \| NK'_N\}$ and $w' = MAC_{AIK_{S_2}}(N \| t')$. S_2 generates $v_2 = h(NK'_N \| R_0)$ and $u_3 = E_{NK_N}\{R_0 \| v_2 \| t' \| w'\}$, $v_3 = MAC_{NIK_N}(S_2 \| N \| u_3)$.

$$S_2 \rightarrow N : S_2 \| N \| u_3 \| v_3$$

3. After verifying v_3, N decrypts u_3 and retrieves R_0, v_2, t' and w'. Then N generates NK'_N and verifies v_2. N generates $v_4 = MAC_{NIK'_N}(N \| S_2 \| \text{ACK} \| R_0 \| R_1)$.

$$N \rightarrow S_2 : N \| S_2 \| \text{ACK} \| v_3$$

4. After verifying v_4, S_2 authenticates N.

3.3. PKI-deployed Protocol

PKI-deployed protocol has a slightly different process from symmetric key based protocol. It uses *Neighbor Sink List*, introduced in section 3.1.2.

3.3.1. Pre-Phase 0: Neighbor Discovery. A sink S_1 generates $v_0 = sign_{sk_1}(h(HELLO \| TS))$, where $HELLO$ is a generic "HELLO" message and TS is the time stamp. S_1 broadcasts v_0 with $HELLO$, TS, pk_1, and $cert_{S_1}$. Because only pk_1 and $cert_{S_1}$ are required for pre-phase1, any nodes that already know pk_1 and $cert_{S_1}$ will ignore this phase when they receive the $HELLO$ message.

3.3.2. Pre-Phase 1: Neighbor Sink List Set Up. When a sink S_2 receives the "HELLO" message of S_1 that has no previous relationship with S_2, S_2 operates as follows:

a. After verifying $cert_{S_1}$ using pk_{BS}, S_2 verifies v_0 using pk_1. Then, S_2 randomly selects R_1, generates $u_1 = enc_{pk_1}\{S_2 \| S_1 \| R_1 \| h(R_1)\}$, and returns u_1 to S_1 as follows.

$$S_2 \rightarrow S_1 : S_2 \| S_1 \| u_1$$

b. S_1 also verifies pk_2 and retrieves R_1 after decryption of u_1. S_1 then randomly selects R_2 and generates $K_{S_1S_2} = KDF(R_1 \| R_2)$. KDF is a key derivation function such as the hash function. S_1 generates $u_2 = enc_{pk_2}\{S_1 \| S_2 \| R_2 \| h(K_{S_1S_2} \| R_2)\}$ and sends u_2 to S_2 as follows.

$$S_1 \rightarrow S_2 : S_1 \| S_2 \| u_2$$

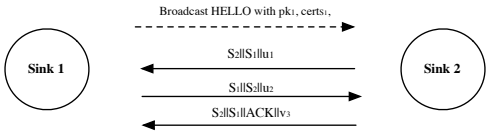

Figure 5: Pre-phase 1: Neighbor Sink List Set Up

c. S_2 decrypts u_2 and retrieves R_2. S_2 also generates $K_{S_1S_2}$ and checks $h(K_{S_1S_2} \| R_2)$ for freshness check. S_2 generates $v_3 = MAC_{K_{S_1S_2}}(S_2 \| S_1 \| ACK \| R_1 \| R_2)$ and sends ACK and v_3 to S_1 as follows.

$$S_2 \rightarrow S_1 : S_2 \| S_1 \| ACK \| v_3$$

d. S_1 verifies v_3 and updates NSL_{S_1}.

As a result, S_1 shares a key $K_{S_1S_2}$ with S_2. The integrity key $IK_{S_1S_2}$ and the cipher key $CK_{S_1S_2}$ are derived from $K_{S_1S_2}$. If a sink receives a "HELLO" message from the sinks in the NSL, ignore this phase. Figure 5 shows a simplified activity diagram of this phase.

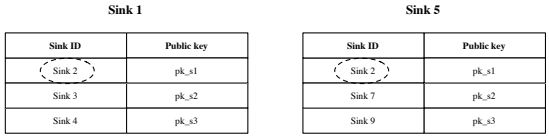

Figure 6: Sink 1 and Sink 5 find the neighbor sink Sink 2 in their Neighbor Sink Lists

3.3.3. Pre-phase 2: Initial Mobile Node Authentication. When a mobile sensor node that is only capable of symmetric key computation is joining the network for the first time, pre-phase 2 is operated as follows:

a. When a node N receives v_0 from S_1, N randomly selects R_1 and generates $u_1 = e_{CK_{N,BS}}\{R_1 \| v_0\}$, $v_1 = MAC_{IK_{N,BS}}(N \| S_1 \| u_1)$, where $CK_{N,BS}$ and $IK_{N,BS}$ are the shared cipher key and integrity key, respectively, between N and BS, and sends u_1 and v_1 to S_1 as follows.

$$N \rightarrow S_1 : N \| S_1 \| u_1 \| v_1$$

b. After receiving u_1 and v_1, S_1 randomly selects R_2, and generates $u_2 = e_{CK_{S_1,BS}}\{u_1 \| R_2\}$ and $v_2 = MAC_{IK_{S_1,BS}}(S_1 \| BS \| N \| u_2 \| v_1)$, where $CK_{S_1,BS}$ and $IK_{S_1,BS}$ are shared the cipher key and integrity key, respectively, between S_1 and BS. Then, S_1 sends u_2, v_1 and v_2 to BS as follows.

$$S_1 \rightarrow BS : S_1 \| BS \| N \| u_2 \| v_1 \| v_2$$

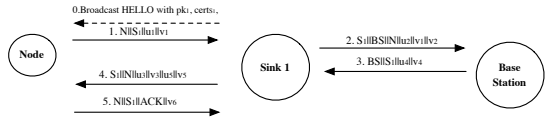

Figure 7: Pre-Phase 2: Initial Mobile Node Authentication

c. After receiving u_2, v_1 and v_2, BS verifies v_2 and decrypts u_2. BS then retrieves R_2 and verifies v_1. BS also retrieves R_1, TS and v_0, and verifies v_0 checking if the TS is valid. BS then generates $u_3 = e_{CK_{N,BS}}\{R_2\}$, $v_3 = MAC_{IK_{N,BS}}(BS \| N \| S_1 \| u_3)$, $u_4 = e_{CK_{S_1,BS}}\{R_1 \| u_3 \| v_3\}$ and $v_4 = MAC_{IK_{S_1,BS}}(BS \| S_1 \| N \| R_2 \| u_4)$, and sends u_4 and v_4 to S_1 as follows.

$$BS \rightarrow S_1 : BS \| S_1 \| u_4 \| v_4$$

d. S_1 verifies v_4 and decrypts u_4. After retrieving R_1, u_3 and v_3, S_1 derives $K_{N,S_1} = KDF(R_1 \| R_2)$ that will be the shared session key between S_1 and N. CK_{N,S_1} and IK_{N,S_1} are the cipher key and integrity key, respectively, derived from K_{N,S_1} individually. S_1 then generates $u_5 = e_{CK_{N,S_1}}\{NSL_{S_1}\}$ and $v_5 = MAC_{IK_{N,S_1}}(S_1 \| N \| R_1 \| u_5)$. S_1 then sends u_3, v_3, u_5, and v_5 to N as follows.

$$S_1 \rightarrow N : S_1 \| N \| u_3 \| v_3 \| u_5 \| v_5$$

e. After verifying v_3, N decrypts u_3 and retrieves R_2. N derives K_{N,S_1} using R_2. N can verify v_5 and decrypt u_5. After decrypting u_5, N obtains NSL_{S_1}. N then generates $v_6 = MAC_{IK_{N,S_1}}(N \| S_1 \| ACK \| R_2 \| R_1)$ and sends ACK and v_6 to S_1 as follows.

$$N \rightarrow S_1 : N \| S_1 \| ACK \| v_6$$

f. After S_1 verifies v_6, S_1 authenticates N.

Figure 9 shows the simplified activity diagram of this phase.

3.3.4. Reconnecting Mobile Node Authentication. When an authenticated node N moves and reconnects to another sink S_2, N and S_2 proceed as follows. Assume N receives v_0 from S_2.

a. N randomly selects R_1, and generates $u_1 = e_{CK_{N,S_1}}\{R_1 \| v_0\}$ and $v_1 = MAC_{IK_{N,S_1}}(N \| S_2 \| t_1 \| v_0)$. N then sends NSL_{S_1}, u_1, and v_1 to S_2 as follows.

$$N \rightarrow S_2 : N \| S_2 \| NSL_{S_1} \| u_1 \| v_1$$

b. S_2 verifies NSL_{S_1} and checks whether S_1 is a neighbor sink of S_2 with the public key pk_{S_1}. If S_1 is a neighbor sink of S_2, S_2 generates $u_2 = e_{CK_{S_1,S_2}}\{N \| R_2 \| u_1 \| v_1\}$ and $v_2 = MAC_{IK_{S_1,S_2}}(S_2 \| S_1 \| u_2)$, where CK_{S_1,S_2} and IK_{S_1,S_2} are derived from K_{S_1,S_2}, and sends u_2 and v_2 to S_1 as follows.

$$S_2 \rightarrow S_1 : S_2 \| S_1 \| u_2 \| v_2$$

c. After verifying v_2, S_1 decrypts u_2, and retrieves N, R_2, u_1, and v_1. S_1 then finds CK_{N,S_1} and IK_{N,S_1} to verify v_1 and decrypt u_1. S_1 generates $u_3 = e_{CK_{N,S_1}}\{R_2\}$, $v_3 = MAC_{IK_{N,S_1}}(S_2 \| R_1 \| u_3)$, $u_4 = e_{CK_{S_1,S_2}}\{R_1 \| u_3 \| v_3\}$ and $v_4 = MAC_{IK_{S_1,S_2}}(S_1 \| S_2 \| N \| u_4 \| R_2)$, and sends u_4 and v_4 to S_2.

$$S_1 \rightarrow S_2 : S_1 \| S_2 \| u_4 \| v_4$$

d. S_2 verifies v_4 and decrypts u_4. S_2 then retrieves R_2, u_3 and v_3. S_2 derives a new session key $K_{N,S_2} = KDF(R_1 \| R_2)$, and also derives CK_{N,S_2} and IK_{N,S_2} individually. S_2 generates $u_5 = e_{CK_{N,S_2}}\{NSL_{S_2}\}$ and $v_5 = MAC_{IK_{N,S_2}}(S_2 \| N \| u_3 \| u_5 \| v_3 \| R_1 \| R_2)$. S_2 then sends u_3, v_3, and v_5 to N.

$$S_2 \rightarrow N : S_2 \| N \| u_3 \| v_3 \| v_5$$

e. N verifies v_3 and decrypts. N generates K_{N,S_2} and verifies v_5 using IK_{N,S_2}. N then generates $v_6 = MAC_{IK_{N,S_2}}(N \| S_2 \| ACK \| R_2 \| R_1)$ and sends ACK and v_6 to S_2 as follows.

$$N \rightarrow S_2 : N \| S_2 \| ACK \| v_6$$

In real environments, sensor nodes may be irregularly distributed. When N requests the connection to S_5, S_5 may not be able to authenticate N properly because S_5 may not be a neighboring sink of S_1. In this case, S_5 can authenticates N via S_2 modifying step 2 as follows.

a. S_5 checks whether S_1 is a neighbor of S_5. If S_1 is not a neighbor of S_5, S_5 finds a common neighboring sink from NSL_{S_1} and NSL_{S_5}. When S_5 has a common neighbor sink S_2 with S_1 as in Figure 8, S_5 randomly selects R_2 and generates $u_2 = e_{CK_{S_2,S_5}}\{N \| R_2 \| NSL_{S_1} \| u_1 \| v_1\}$ and $v_2 = MAC_{IK_{S_2,S_5}}(S_5 \| S_2 \| u_2)$. S_2 then sends u_2 and v_2 to S_2.

$$S_5 \rightarrow S_2 : S_5 \| S_2 \| u_2 \| v_2$$

b. S_2 verifies v_2 and decrypts u_2. S_2 then verify NSL_{S_1} and checks whether S_1 is a neighbor sink of S_2. If S_1 is a neighbor sink of S_2, S_2 generates $u_3 = e_{CK_{S_1,S_2}}\{N \| R_2 \| u_1 \| v_1\}$ and $v_3 = MAC_{IK_{S_1,S_2}}(S_2 \| S_1 \| u_3)$, and sends u_3 and v_3 to S_1 as follows.

$$S_2 \rightarrow S_1 : S_2 \| S_1 \| u_3 \| v_3$$

Also, step 4 is modified such that S_1 sends the authenticating information to S_5 via S_2. We omit the details of the modified step, since the procedure is similar to the step a and b.

3.4. How to Reduce Overhead in Initial Authentication

While our protocols in [7, 8] could reduce overhead in re-authentication, they still have the same computation and communication overheads for initial authentication.

In order to reduce overhead of sensor nodes during initial authentication, we also proposed authentication protocol for 3G-WSN integrated networks [9]. The main idea is that the application of mobile sensor nodes such as RF4CE [19] will follow integration of multiple networks. Thus the use of the overwhelming capabilities of 3G networks (or 4G eventually) could reduce the use of communication within WSN for establishing secure channel.

3.5. Analysis of Protocols

Detailed analyses of protocols are shown in [7, 8]. We do not show the detail in this paper.

4. Related Work

In this section, we briefly review well-known key agreement protocols designed for the sensor networks. As a commercial solution, Zigbee [3] specifies the key agreement architecture by key pre-distribution. In their architecture, each node pre-installs its unique keys that are shared with other entities and the network key that is shared with the entire network by the manufacturer. In order to support mobility of a node using a unique key, each node must contain as many keys as the number of nodes. Thus, most studies on authenticated key agreements attempt to increase the efficiency.

4.1 Authenticated Key Agreement Protocols for Static Sensor Networks

Most previous key agreement protocols were based on the symmetric key cryptosystem. Eschenauer and Gligor proposed the pairwise key agreement protocols based on the random key pre-distribution [5] that enabled the sharing of pairwise keys from the pre-distributed key pool. In their protocol, each node stores m number of keys selected from a key pool in the initial stage. After the nodes are deployed, each node shares the key information with its neighboring nodes. When the shared keys are found, the node establishes secure links between the sinks that share the keys. After a link is established, both nodes generate a pairwise key for secure communication. However, the network establishment has a probability of failure that is increased in the case of irregular deployment of sensor nodes or unpredictable interruptions.

Zhu *et al.* [18] introduced the group-key-based key agreement model that minimized threats of compromised nodes. Every node has a unique key, pairwise keys with neighboring nodes, a cluster key shared with all neighboring nodes, and a global key shared with the entire network. However, they assumed that the networks are static.

Abraham and Ramanatha [1] proposed an authentication and initial shared key establishment model in hierarchical clustered networks. Ibriq and Mahgoub [11] proposed an efficient model that deployed a "partial key escrow table" for sinks. Using the key escrow table, a sink can self-generate a shared key for the attached nodes. However, all sinks have to maintain the information of every node in the table to support node mobility.

4.2 Authenticated Key Agreement Protocols For Dynamic WSN

4.2.1 Distributed Authentication Model. Fantacci et al. [8] proposed the distributed node authentication model that does not require the base station to act as the centralized authenticator (CA), as shown in

Figure 2. In their model, every node shares partial authentication information of other nodes based on the secret sharing scheme [16]. A node sends an authentication request to another node; e.g., the node N_2 is the authenticator and other nodes such as N_5 and N_6 are distributed authentication servers. The overhead on all nodes in this model is large due to their involvement in the authentication process. Since each node has to participate in the authentication procedures as an authenticator or as an authentication server, the computational and communication overhead would significantly increase as a result of frequent authentication requests. Once a node N_1 is authenticated by N_2, as shown in Figure 7 (a), N_1 sends authentication requests to N_7, as shown in Figure 7 (b). In the figure, N_3, N_4, N_5, and N_6 are involved in both authentication processes as authentication servers.

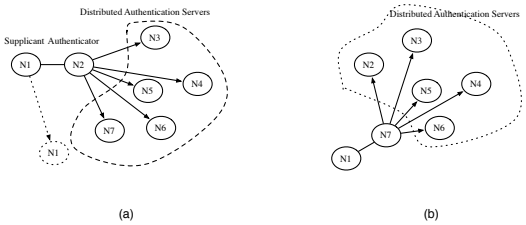

Figure 5: Fantacci's Distributed Node Authentication Model (a) Initial authentication by N_2 (b) N_1 reauthenticated by N_7

4.2.2 PKI-based Model. Although PKI brings strong and advanced security services, most studies focused on the symmetric key cryptosystem-based approach, due to the insufficient computational resources for PKI of the sensor nodes. However, many efforts that enable PKI for sensor networks such as TinyPK [17] and TinyECC [15] are often proposed.

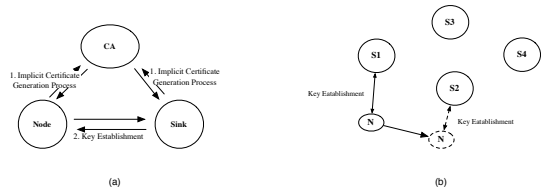

Figure 6: (a) Simplified representation of the processes of Huang's Key Agreement Model (b) Applying Huang's model in Dynamic Sensor Network

Huang *et al.* proposed a self-organizing algorithm by using Elliptic Curve Cryptography (ECC) [10]. Huang's model has two phases: *Implicit certificate generation process* and *Hybrid key establishment process*. Once the certificates are issued to nodes, they can self-establish the pairwise keys by exchanging the certificates with other sinks. Simplified representation of the processes is shown in Figure 9 (a). Although Huang *et al.* did not state that their protocol could be applicable to dynamic WSN, their protocol can support node re-authentication. After the certificate is issued to the node *N*, *N* is authenticated by a sink S_1. When *N* moves and requests the re-authentication to another sink S_2, S_2 can easily authenticate *N* again as shown in Figure 9 (b).

However, their model has two critical problems: (1) all sensor nodes must contact the CA to obtain their certificates. (2) Direct contact is required between each sensor node (including mobile sensor node) and the CA, which is not considered practical for large-scale networks. If an implicit certificate is pre-installed to every sink, the advantage of the protocol may be significantly reduced. The other is that every node has to be capable of ECC computation. Even though PKI-based applications for sensor networks will be available in the near future with efficient implementations, the public key-based security architecture still requires more advanced computational power and resources. A sensor node that is only capable of a lightweight cryptosystem such as AES or SHA-1 may not be able to connect to such networks.

5. Conclusion

While most current security protocols are designed for the static sensor networks and has several problems that applying such models in dynamic environment may occur significantly large resource drain.

In this paper, we introduced our efficient model for authenticated key agreement in dynamic WSN and showed several protocols we proposed in [7, 8]. We showed symmetric key based protocol and hybrid protocol that combines symmetric key base model with PKI based model.

Our protocols enable the reduced authentication process for the mobile node and can be used in various application of WSN.

References

[1] Jibi Abraham and K S Ramanatha. An Efficient Protocol for Authentication and Initial Shared Key Establishment in Clustered Wireless Sensor Networks. Proceeding of Third IFIP/IEEE International Conference on Wireless and Optical Communications Networks, 2006.

[2] Ian F. Akyildiz and Ismail H. Kasimoglu. Wireless sensor and actor networks: research challenges. Ad Hoc Networks, 2(4):351 - 367, 2004.

[3] William C Craig. Zigbee:Wireless Control That Simply Works. Zigbee Alliance, 2005.

[4] Jeremy Elson and Deborah Estrin. Random, Ephemeral Transaction Identifiers in Dynamic Sensor Networks. 21st International Conference on Distributed Computing Systems, :pp. 0459, 2001.

[5] L. Eschenauer and V.D. Gligor. A key management scheme for distributed sensor networks". in Proceedings of the 9th ACM conference on Computer and Communications Security (CCS). Washington. DC. USA, :41-47, 2002.

[6] Gill, K. and Shuang-Hua Yang and Fang Yao and Xin Lu. A Zigbee-based home automation system. IEEE Transactions on Consumer Electronics, 55(2):422-430, 2009.

[7] Kyusuk Han and Kwangjo Kim and Taeshik Shon. Untraceable Mobile Node Authentication in WSN. Sensors, 10(5):4410-4429, 2010.

[8] Kyusuk Han and Taeshik Shon and Kwangjo Kim. Efficient Mobile Sensor Authentication In Smart Home and WPAN. IEEE Trans. on Consumer Electronics, 56(2):591-596, 2010.

[9] Kyusuk Han, Kwangjo Kim, Wook Choi, Hyohyun Choi, Jungtaek Seo, Taeshik Shon, Efficient Authenticated Key Agreement Protocols for Dynamic Wireless Sensor Networks, Ad Hoc & Sensor Wireless Networks, Volume 12 , Number 3 , Mar 2012, pp , ISSN: 1551-9899

[10] Qiang Huang and Johnas Cukier and Hisashi Kobayashi and Bede Liu and Jinyun Zhang. Fast authenticated key establishment protocols for self-organizing sensor networks. WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications, 2003.

[11] J. Ibriq and I. Mahgoub. A Hierarchical Key Establishment Scheme for Wireless Sensor Networks. Proceedings of 21st International Conference on Advanced Networking and applications (AINA'07), :210-219, 2007.

[12] C. Karlof and D. Wagner. Secure routing in wireless sensor networks. In Proc. of SNPA'03, Anchorage, Alaska, :113-127, 2003.

[13] Chris Karlof and Naveen Sastry and David Wagner. TinySec: a link layer security architecture for wireless sensor networks. SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems, 2004.

[14] HangRok Lee and YongJe Choi and HoWon Kim. Implementation of TinyHash based on Hash Algorithm for Sensor Network. Proceedings of World Academy of Science, Engineering and Technology, 10:135-139, 2005.

[15] An Liu and Peng Ning. TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks. 2008 International Conference on Information Processing in Sensor Networks, 2008.

[16] Adi Shamir. How to share a secret. Communications of the ACM, 22(11):612--613, 1979.

[17] Ronald Watro and Derrick Kong and Sue-fen Cuti and Charles Gardiner and Charles Lynn and Peter Kruus. TinyPK: Securing sensor networks with public key technology. Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks, :59-64, 2004.

[18] Zhu, Sencun and Setia, Sanjeev and Jajodia, Sushil. LEAP+: Efficient security mechanisms for large-scale distributed sensor networks. ACM Trans. Sen. Netw., 2(4):500--528, 2006.

[19] ZigBee alliance. RF4CE Standard Specification Release 1.0. 2009.

