KALwEN+: Practical Key Management Schemes for Gossip-Based Wireless Medical Sensor Networks

Zheng Gong[1], Qiang Tang[1], Yee Wei Law[2], and Hongyang Chen[3]

[1] Faculty of EWI, University of Twente, The Netherlands
{z.gong, q.tang}@utwente.nl
[2] Department of EEE, The University of Melbourne, Australia
yee.wei.law@gmail.com
[3] Institute of Industrial Science, The University of Tokyo, Japan
hongyang@mcl.iis.u-tokyo.ac.jp

Abstract. The constrained resources of sensors restrict the design of a key management scheme for wireless sensor networks (WSNs). In this work, we first formalize the security model of ALwEN, which is a gossip-based wireless medical sensor network (WMSN) for ambient assisted living. Our security model considers the node capture, the gossip-based network and the revocation problems, which should be valuable for ALwEN-like applications. Based on Shamir's secret sharing technique, we then propose two key management schemes for ALwEN, namely the KALwEN+ schemes, which are proven with the security properties defined in the security model. The KALwEN+ schemes not only fit ALwEN, but also can be tailored to other scalable wireless sensor networks based on gossiping.

Keywords: Wireless medical sensor network, Gossiping, Key management.

1 Introduction

Following the improvement of wireless technologies and embedded systems, the potential of wireless sensor networks (WSNs) for various applications has been drawing a great deal of attention from the academia and the industry. For WSNs, one of the promising applications is healthcare. A wireless medical sensor network (WMSN, sometimes also called body sensor network) [19], which can be developed from a WSN, is a developing technology for long term monitoring of biological events or any abnormal condition of patients for realizing Ambient Assisted Living (AAL) [1]. In general, a WMSN is a moderate-scale wireless network of low-cost sensors. The purpose of WMSN is to monitor the user's physiological parameters and the related information in environment, e.g., ECG, EMG, EEG, SpO_2 and blood pressure. The collected data will be sent to doctors or nurses for daily diagnosis. A typical scenario of WMSN is illustrated in Figure 1.

X. Lai, M. Yung, and D. Lin (Eds.): Inscrypt 2010, LNCS 6584, pp. 268–283, 2011.

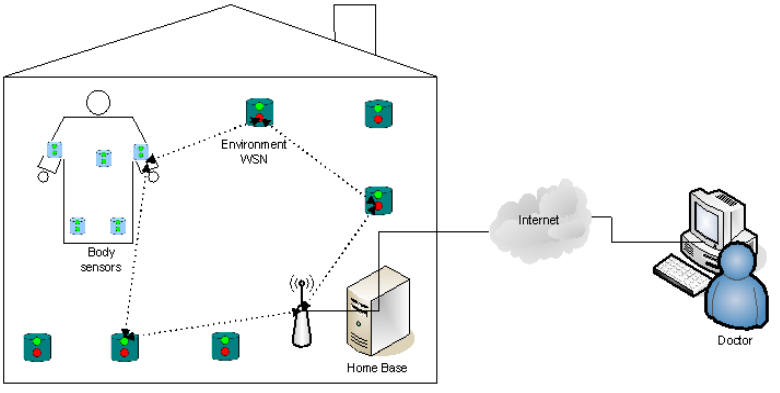

Fig. 1. A Scenario of Wireless Medical Sensor Network

In practice, sensors used in WMSNs also have limited computational abilities and small memories, typically with a low-end CPU and RAM in KBytes level. These factors are important not only in the implantable but also in the external sensor settings because they determine how "hidden" and "pervasive" the sensors are. A gossip protocol is a style of computer-to-computer communication protocol inspired by the form of gossip seen in social networks. Since gossip-based network protocol is proven to be energy-efficient, it would be a low-cost candidate for realizing a WMSN via gossiping [10]. Recently, the ALwEN project [2] built a gossip-based wireless sensor network with 1000 nodes. The estimated lifetime of the network can be 1-2 years, which is a promising property in practice.

Although gossip-based WSN is energy-efficient, designing a appropriate key management scheme for WMSN is a challenging task. In the gossip mode, each node will send out messages to 1-hop neighbor nodes with a well-chosen probability. Thus the security model should consider the situations that all nodes can receive the message, and the message might be dropped during multi-hops. Moreover, the security and privacy problems related to healthcare systems are critical [3]. As a recent study has demonstrated, medical devices that do not support any confidentiality and authentication function are prone to eavesdropping and attacks [11]. Basically, solving these problems requires a key management scheme, which handles the cryptographic keys in a right manner, to provide data confidentiality and authenticity. In the literature, many key management schemes have been proposed for broadcast/gossip WSNs [8,13,16]. However, a WMSN-oriented key management must consider the following differences. Firstly, in WMSN applications, nodes might be added or removed frequently. For the ease of a user, the initialization or revocation of such nodes should be designed as agile as possible. Since we suppose the added/removed nodes might be tampered, the resilience of compromise becomes serious in WMSN key management. Secondly, a typical WMSN is a moderate-scale WSN, so probabilistic key sharing schemes that are designed for large-scale WSNs are not suitable [6,7,12]. For practical

applications, a good WMSN key management scheme must consider the above differences carefully, whilst balancing the applicability and the security.

Recently, Law et al. propose a novel WMSN key management scheme, which is called KALwEN [14]. But KALwEN relies on a smart Faraday cage and unicast communication channels, which might be impractical in some cases. In this work, our main contribution are two new key management schemes, namely the KALwEN+ schemes, which are secure against active and aggressive adversaries respectively. Compared to KALwEN, KALwEN+ does not require a Faraday cage, and the communication can be fully broadcast for satisfying gossip-based networks. Based on Shamir's secret sharing technique, KALwEN+ schemes support an efficient way to add/remove nodes. Using formal analysis, we prove that the KALwEN+ schemes are secure in our formalized security model. Based on their theoretical performances, the KALwEN+ schemes not only fit ALwEN, but also can be tailored to other scalable wireless sensor networks based on gossiping.

The rest of this paper is organized as follows. In Section 2, we first describe the system environment, then define the security model for KALwEN+. In Section 3, we describe the KALwEN+ scheme secure against active adversaries and prove its security in our security model. In Section 4, we describe the KALwEN+ scheme secure against aggressive adversaries and prove its security in our security model. In Section 5, we present the performance analysis for KALwEN+ schemes. In Section 6, we conclude the paper.

2 Key Distribution Schemes for Gossip-Based WMSN

In this section we first describe the system environment, then formulate the security properties of key distribution schemes which are specifically tailored to gossip-based WMSN. The security formulations follow that of Bellare and Rogaway [4].

2.1 Environment of Gossip-Based WMSN

Due to the special setting of gossip-based WMSN as shown in Figure 2, at the beginning of the key distribution, a node denoted as the *sink node* is connected to trusted device Dev (e.g., a home-based computer) and key distribution messages will be broadcast by the sink node as an initiator. Then, the sink node and other nodes will engage in a key management scheme. The resultant session keys will be used to protect the data collection and the gossip communications.

2.2 Description of Key Distribution Schemes

We consider an environment which can consist of maximal N sensor nodes, say $node_i$ $(1 \leq i \leq N)$, and a trusted device Dev, such as a PC or a programmer or any other trusted infrastructure, which serves as a fully trusted third party (TTP). All nodes are honest and follow the pre-configured instructions, unless they are compromised by an adversary. In addition, we note that the trusted device Dev typically does not have the ability to connect to any node through

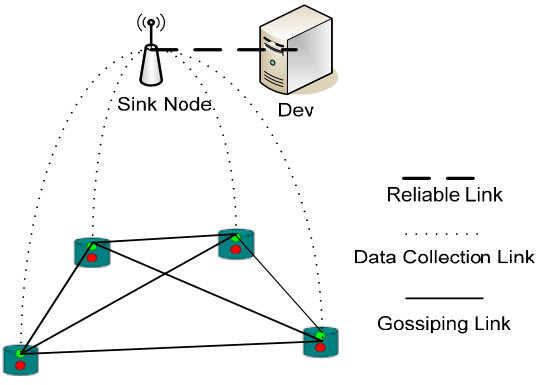

Fig. 2. Environment of Gossip-based WMSN

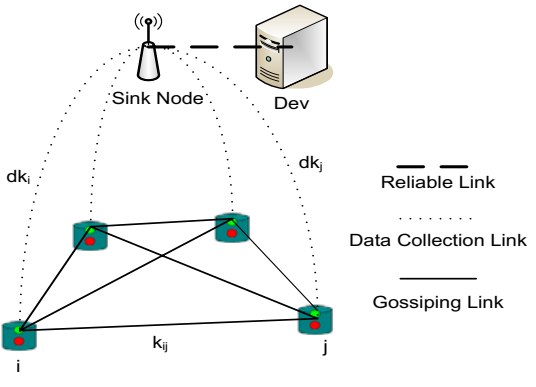

Fig. 3. Key Distribution of WMSN

wireless communication. To facilitate the establishment of our security model, we assume that a key distribution scheme for gossip-based WMSN consists of the following three phases.

1. *System setup.* In this phase, the trusted device Dev generates the long-term credentials. In the symmetric-key setting, a global key k_G is generated, while in the public-key setting a public/private key pair (PK_G, SK_G) is generated. In addition, the trusted device Dev generates some public system parameters $params$.
2. *Node setup.* In this phase, every node $node_i$ is initialized by the trusted device Dev. In the symmetric-key setting, the global key k_G is stored in the node. In the public-key setting, the trusted device Dev generates a public/private key pair (PK_i, SK_i) and stores $(PK_G, Cert_i, PK_i, SK_i, params)$ in $node_i$, where $Cert_i$ is a certificate of PK_i generated with PK_G.

Note that the above two steps can be executed outside the key distribution scheme. The manufacturer can generate the certificates and the global key, and then distribute them to the trusted device and the nodes beforehand.

3. *Key distribution.* In this phase, the following two types of session keys will be distributed to a group of nodes, say $node_i$ $(1 \leq i \leq N')$ and $N' \leq N$.
   - The first type is *data collection keys* used for data collection. For node $node_i$, the data collection key is denoted as dk_i. This key is used for end-to-end communication between $node_i$ and the data collection gateway (namely, the sink node).
   - The second type is *pairwise keys* used for nodes to securely communicate with each other. For a pair of nodes $node_i$ and $node_j$, the the pairwise key is denoted as $k_{i,j}$.

   In addition, we assume that the trusted device Dev keeps a counter *ctr* to count all the key distribution sessions. Identified by the counter *ctr*, we denote an invocation of the key distribution protocol as a *session.*

2.3 Security Properties and Their Formulations

In our security model, we only consider attacks from adversaries, whose main focus is to obtain information about the session keys, including cluster keys and pairwise keys, in a certain session. We make the following assumptions:

1. No adversary is present in the *system setup* and *node setup* phases, so that no information about the long-term credentials will be leaked in both phases.
2. An adversary may mount a denial of service (DoS) attack against the key distribution process. How to make a key distribution scheme secure in this case is beyond the scope of our model.

With respect to the secrecy of the data collection keys and pairwise keys, we consider the following types of adversaries.

- *Passive Adversary* ($\mathcal{A}^-$). This type of adversary can only passively eavesdrop on the wireless communications in the network.
- *Active Adversary* ($\mathcal{A}$). This type of adversary can not only eavesdrop on, but also manipulate the wireless communications in the network. The possible manipulation of communication includes delaying, deleting, inserting, and replacing messages.

It is worth noting that both types of adversaries are outsiders since we assume all nodes are honest. In addition, since active adversaries are more powerful than the passive ones, a scheme secure against the former will also be secure against the latter.

Following the work by Bellare and Rogaway [4], the security of a key distribution scheme for gossip-based WMSN is evaluated by the attack game between a

1. Setup: the challenger generates the parameters for the trusted device Dev and publishes the public parameters.
2. Phase 1: Besides delivering messages for all sessions, the adversary is allowed to issue the following types of queries.
   (a) $\mathsf{Invoke}(set, node_i)$: The trusted device Dev initiates a new session to distribute cluster keys and pairwise keys to the nodes in the set set which is a subset of $\{node_j | 1 \leq j \leq N\}$. The node $node_i$ belongs to the set set and acts as the sink node.
   (b) $\mathsf{Corrupt_k}(ctr, node_j)$: If the session identified by ctr has successfully ended and $node_j$ has been involved in the session, the challenger sends the data collection key and pairwise keys of $node_j$ to the adversary. Otherwise, the challenger returns nothing.

   At some point, the adversary chooses a counter value ctr^* and a user index j, such that, in the session identified by ctr^*, $node_j$ has successfully ended with $dk_j^*, pk_{j,t}^*$ for all t such that $node_t$ is also involved in the session. This is subject to the restriction that there has been no $\mathsf{Corrupt_k}(ctr^*, node_t)$ query for any t.
3. Challenge: Select $b \in_R \{0, 1\}$. If $b = 0$, send $dk_j^*, pk_{j,t}^*$ for all t such that $node_t$ is also involved in the session, otherwise send a replacement to the adversary, where the keys are replaced by a set of random values.
4. Phase 2: The adversary is allowed to issue the same types of queries as in Phase 1, and is subject to the same restriction. At some point, the adversary terminates by outputting a guess bit b'.

Fig. 4. The Attack Game

challenger and an adversary, as shown in Fig. 4, where the adversary's advantage is defined to be $|\Pr[b = b'] - \frac{1}{2}|$. It is worth noting that the challenger faithfully simulates all these activities of the trusted device Dev and all the nodes.

Definition 1. *A key distribution scheme for gossip-based WMSN is secure against (passive and) active adversaries, if any polynomial-time adversary has only negligible advantage in the attack game defined in Fig. 4.*

It is worth stressing that in the attack game defined in Fig. 4, the adversary is allowed to obtain all data collection keys and pairwise keys in all sessions except ctr^*. As a result, a secure scheme under this definition achieves known-key security [15].

Compared with other settings, in gossip-based WMSN, it is reasonable to assume that it is very difficult for an adversary to physically capture the nodes since they will be locked indoor or worn by patients. In other words, key distribution schemes secure against passive and active adversaries provide adequate security guarantees in most application scenarios. However, in some scenarios, higher security level may be required in the presence of an *aggressive adversary* $\mathcal{A}^+$. Besides eavesdropping on and manipulating wireless communications, this type of adversary is also capable of physically compromising some wireless nodes in the network even before the key management.

1. Setup: the challenger generates the parameters for the trusted device Dev and publishes the public parameters.
2. Phase 1: Besides delivering messages for all sessions, the adversary is allowed to issue the following types of queries.
   (a) $\mathsf{Invoke}(set, node_i)$: The trusted device Dev initiates a new session to distribute cluster keys and pairwise keys to the nodes in the set set which is a subset of $\{node_j | 1 \leq j \leq N\}$. The node $node_i$ belongs to the set set and acts as the sink node.
   (b) $\mathsf{Corrupt_k}(ctr, node_j)$: If the session identified by ctr has successfully ended and $node_j$ has been involved in the session, the challenger sends the data collection key and pairwise keys of $node_j$ to the adversary. Otherwise, the challenger returns nothing.
   (c) $\mathsf{Corrupt_l}(index)$: The challenger returns the long-term public/private keys of $node_{index}$ to the adversary.

   At some point, the adversary chooses a counter value ctr^* and a user index j, such that, in the session identified by ctr^*, $node_j$ has successfully ended with $dk_j^*, pk_{j,t}^*$ for all t which satisfies that $node_t$ is also involved in the session. This is subjected to the following restrictions.
   (a) Suppose the node $node_i$ is the sink node in the session identified by ctr^*. There has been no $\mathsf{Corrupt_l}(i)$ and $\mathsf{Corrupt_k}(ctr^*, node_i)$ queries. The requirement also applies to $node_j$. Note that the adversary may choose $j = i$ in the challenge.
   (b) Suppose set^* is the set of nodes in the session identified by ctr^* satisfying that if $node_j \in set^*$ then there has been no $\mathsf{Corrupt_k}(ctr^*, node_j)$ query and no $\mathsf{Corrupt_l}(j)$ query. The size of set^* is at least 2.
   (c) In the session identified by ctr^*, at most $t-1$ nodes have been issued a $\mathsf{Corrupt_k}$ query.
3. Challenge: Select $b \in_R \{0, 1\}$. If $b = 0$, send $dk_j^*, pk_{j,t}^*$ for all t which satisfies that $node_t$ is also involved in the session and there has been no $\mathsf{Corrupt_k}(ctr^*, node_t)$ query and and no $\mathsf{Corrupt_l}(t)$ query, otherwise send a replacement to the adversary, where the keys are replaced by a set of random values.
4. Phase 2: The adversary is allowed to issue the same types of queries as in Phase 1, with the following restriction.
   (a) There has been no $\mathsf{Corrupt_k}(ctr^*, node_h)$ query for any h satisfying that there has been no $\mathsf{Corrupt_k}(ctr^*, node_h)$ query in Phase 1.

   At some point, the adversary terminates by outputting a guess bit b'.

Fig. 5. The Enhanced Attack Game

The security against an aggressive adversary is evaluated by the attack game between a challenger and an adversary, as shown in Fig. 5, where the adversary's advantage is defined to be $|\Pr[b = b'] - \frac{1}{2}|$.

Definition 2. *A key distribution scheme for WMSN is secure against an aggressive adversary, if any polynomial-time adversary has only negligible advantage in the attack game defined in Fig. 5.*

It is worth stressing that in the attack game defined in Fig. 5, the adversary is allowed to obtain all data collection keys and pairwise keys in all sessions except ctr^*, and it is also allowed to obtain all long-term private keys of all nodes in Phase 2. As a result, a secure scheme under this definition achieves known-key security and perfect forward security [15].

3 Scheme Secure against Active Adversaries

In this section, we propose a key distribution scheme which is secure against active adversaries. In this scheme we use symmetric key cryptographic primitives, including message authentication code (MAC) algorithms [15] and symmetric key encryption schemes. We make use of Shamir's secret sharing scheme [17] to deal with the issues such as adding nodes and key recovery in emergency situations.

3.1 Preliminaries

A MAC algorithm is a family of functions $\{\mathsf{MAC}_k\}$, parameterised by a secret key k, with the following properties:

1. Ease of computation: for a known function MAC_k, given a value k and an input x, $\mathsf{MAC}_k(x)$ is easy to compute. This result is called the MAC-value or MAC.
2. Compression: MAC_k maps an input x of arbitrary finite bit-length to an output $\mathsf{MAC}_k(x)$ of fixed bit-length.

Definition 3. *A MAC algorithm is said to be secure against existential forgery if, for any fixed key k (not known to the attacker), and given any number of MAC queries $\mathsf{MAC}_k(x)$, where the values of x may be chosen by the adversary after observing the results of previous queries, a adversary can only succeed with a negligible probability in finding a pair $(x^*, \mathsf{MAC}_k(x^*))$ where x^* (which could be chosen by the attacker) was not in the set of MAC queries.*

Shamir's secret sharing scheme [17] is based on the polynomial interpolation: given k points $(x_1, y_1), (x_2, y_2), \cdots, (x_k, y_k)$, where all elements are from a finite field $\mathbb{F}$ and x_i $(1 \leq i \leq k)$ are distinct, there is one and only one polynomial $f(x)$ of degree $k-1$ such that $f(x) = y_i$ for all is. To hide a secret d, first pick a random $k-1$ degree polynomial $f(x) = d + a_1 x + \cdots + a_{k-1}x^{k-1}$ and sets $d_j = f(j)$ for $1 \leq j \leq n$ where $n \geq k$. It is straightforward to verify that, given any subset of k tuples of the set $\{(i, d_i) | 1 \leq i \leq n\}$, we can find the coefficients of $f(x)$ by interpolation and then obtain $d = f(0)$. Given just $k-1$ of these values, d is indistinguishable from a random element from $\mathbb{F}$.

Let $\mathsf{F} : K \times D \to R$ be a function family, where $K = \{0,1\}^x, D = \{0,1\}^y, R = \{0,1\}^z$ for some integers x, y, z. F is said to be a pseudorandom function family if, given the input-output behaviors, an adversary can only distinguish $\mathsf{F}(k, \cdot)$ from Ran with a negligible probability, where k is randomly chosen from $\{0,1\}^x$ and $\mathsf{Ran} : D \to R$ is a random function [9].

3.2 Description of the Scheme

In the *system setup* phase, the trusted device Dev selects a symmetric encryption algorithm $(\mathsf{ENC}, \mathsf{DEC})$, an MAC algorithm MAC, and a symmetric key $k_G = (k_1, k_2)$. It also choose a finite field $\mathbb{F}$ for Shamir's secret sharing.

In the *node setup* phase, $(k_G, \mathbb{F})$ is stored in the node. For simplicity, we assume all nodes have been programmed to perform all the operations in the key distribution scheme. The key distribution scheme is as follows.

1. A node $node_i$, which is connected to the trusted device Dev, becomes a sink node, broadcasts a bootstrap message to the network. The bootstrap message is defined as follows

$$node_i + \mathsf{Dev} \rightarrow * : ctr,\ \mathsf{ENC}_{k_1}(k_s),\ \mathsf{MAC}_{k_2}(1||ctr||\mathsf{ENC}_{k_1}(k_s)), \quad (1)$$

   where k_s is a randomly-chosen ephemeral key for MAC.
2. After receiving the message, if the value of ctr is smaller than the local counter value, $node_j$ terminates by broadcasting a failure message. Otherwise, it sets the local counter value to be ctr, decrypts $\mathsf{ENC}_{k_1}(k_s)$, and checks

$$\mathsf{MAC}_{k_2}(1||ctr||\mathsf{ENC}_{k_1}(k_s)).$$

   If the MAC code is correct, it sends $(n_j, \mathsf{MAC}_{\mathsf{H}(1||k_s)}(2||ctr||ID_j||n_j)$ to the sink node, where n_j is a nonce.

$$node_j \rightarrow node_i : n_j, \mathsf{MAC}_{\mathsf{H}(1||k_s)}(2||ctr||ID_j||n_j). \quad (2)$$

3. After receiving the message from $node_j$, the sink node first checks the MAC code $\mathsf{MAC}_{\mathsf{H}(1||k_s)}(2||ctr||ID_j||n_j)$. If the check fails, it terminates by broadcasting a failure message. Otherwise, it continues. *At a certain point, the sink node learns that session keys need to be distributed to a group of nodes, say* $node_j$ $(1 \le j \le N')$ *and* $N' \le N$. The sink node computes an ephemeral key pool $\Gamma = \{ek_1, ek_2, \cdots, ek_{N'}, ek'_1, ek'_2, \cdots, ek'_{N'}\}$, where $1 \le j \le N', j \ne i$

   (a) Using Shamir's (t, N)-threshold secret sharing technique, generate N shares $\{(j, sh_j)|1 \le i \le N\}$ to hide a secret $r \in_R \mathbb{F}$.

   (b) Send the following message to the node $node_j$

$$node_i \rightarrow node_j : \mathsf{ENC}_{\mathsf{H}(2||ID_j||k_s)}(j||sh_j||sk_j||T_j),\\ \mathsf{MAC}_{k_2}(ID_j||n_j||ctr||\mathsf{ENC}_{\mathsf{H}(2||ID_j||k_s)}(j||sh_j||sk_j||T_j)), \quad (3)$$

   where $sk_j = \mathsf{H}(3||ctr||ID_j||r)$ and T_j is a concatenation of $pk_{t,j}$ for all t such that $ek_t \in \Gamma$ and $t \ne j$. $pk_{t,j}$ is set to be $\mathsf{H}(4||ctr||ID_t||ID_j||r)$ if $t < j$, and $\mathsf{H}(4||ctr||ID_j||ID_t||r)$ otherwise. Consequently, $pk_{t,j} = pk_{j,t}$ holds.

   (c) The sink nodes stores r and the shares $\{(j, sh_j)|N' + 1 \le i \le N\}$ at the trusted device Dev.

4. After receiving the message, $node_j$ first checks the MAC code. If the check fails, it terminates by broadcasting a failure message. Otherwise, it decrypts $\mathsf{ENC}_{\mathsf{H}(2||ID_j||k_s)}(j||sh_j||sk_j||T_j)$ to obtain the data collection key sk_j, pairwise keys T_j, and the share (j, sh_j). It also updates ctr to be $ctr + 1$.

Lemma 1. *The proposed scheme is secure under Definition 1 given that the MAC algorithm is secure against existential forgery, the encryption algorithm is a pseudorandom function, and* H *is a random oracle.*

Proof sketch. Suppose that an adversary has the advantage ϵ the attack game shown in Fig. 4. We first have the following observation, which implies the integrity of messages received by all nodes (the adversary is not able to manipulate the messages without being detected by some users).

> **Observation**. *During the attack game, in the session identified by ctr^* (and in any other sessions), $node_j$, for any j such that $node_j$ is involved in the session, is supposed to receive the following values:*
>
> $$ctr^*,\ \mathsf{ENC}_{k_1}(k_s),\ \mathsf{MAC}_{k_2}(1||ctr^*||\mathsf{ENC}_{k_1}(k_s)),$$
>
> $$\mathsf{ENC}_{\mathsf{H}(2||k_s)}(j||sh_j||sk_j||T_j),$$
>
> $$\mathsf{MAC}_{k_2}(ID_j||n_j||ctr^*||\mathsf{ENC}_{\mathsf{H}(2||k_s)}(j||sh_j||sk_j||T_j)),$$
>
> *If $node_j$ accepts the values, the probability that these values are not generated (or, simulated) by the challenger is negligible. Intuitively, the reason is that, in the proposed scheme, only sink nodes will generate messages in these format, and based on the existential forgeability of the MAC algorithm an adversary can only forge such messages with a negligible probability. The proof is straightforward so that we skip it here.*

The rest of the security proof is done through a sequence of games [18].

Game_0: In this game, the challenger faithfully simulates the protocol execution and answers the oracle queries from $\mathcal{A}$. Let $\delta_0 = \Pr[b' = b]$, as we assumed at the beginning, $|\delta_0 - \frac{1}{2}| = \epsilon$.

Game_1: The challenger performs faithfully as in Game_0, except that the challenger stops if the values described in the above observation are not generated by the challenger (referred to as the event Ent_1). Let $\delta_1 = \Pr[b' = b]$ at the end of this game. From the Difference Lemma in [18], we have $|\delta_1 - \delta_0| \leq \Pr[Ent_1]$ which is negligible.

Game_2: The challenger performs faithfully as in Game_1, except that, in the session identified by ctr^*, in step 3 of the scheme the messages sent to $node_j$, for any j such that $node_j$ is involved in the session, are replaced with the following, where Ran_j is random function.

$$\mathsf{Ran}_j(j||sh_j||sk_j||T_j),$$

$$\mathsf{MAC}_{k_2}(ID_j||n_j||ctr||\mathsf{Ran}_j(j||sh_j||sk_j||T_j)),$$

Since H is a random oracle and the encryption algorithm is a pseudorandom function, Game_2 is identical to Game_1 unless the adversary queries H with $*||k_s||*$ (referred to as the event Ent_2), where $*$ can be any string. Furthermore, since the encryption algorithm is a pseudorandom function, $\Pr[Ent_2]$ is negligible. Let $\delta_2 = \Pr[b' = b]$ at the end of this game. From the Difference Lemma in [18], we have $|\delta_2 - \delta_1| \leq \Pr[Ent_2]$ which is negligible.

In Game_2, since the encryption of the session keys and shares is provided by random functions, the probability $\delta_2 = \frac{1}{2}$. As a result, we have

$$\begin{aligned}\epsilon &= |\delta_0 - \frac{1}{2}| \\ &\leq |\delta_1 - \delta_0| + |\delta_2 - \delta_1| + |\delta_2 - \frac{1}{2}| \\ &\leq \Pr[Ent_1] + \Pr[Ent_2]\end{aligned}$$

Since $\Pr[Ent_1]$ and $\Pr[Ent_2]$ are negligible, the lemma now follows. □

3.3 Further Remarks

If a key distribution execution has been carried out for $node_j$ $(1 \leq j \leq N')$, later on $node_v$ for any $N' + 1 \geq v \geq N$ may need to join the communications. With respect to the key distribution scheme, there are two possibilities to add a new node into a group. Note the fact that $node_v$ should have been initialized and share the key K_G with the trusted device Dev.

In the first case, if Dev is available, then it can just generate the corresponding data collection key and pairwise keys for $node_v$ based on the secret value r and sends these keys and a share (v, sh_v) to $node_v$ through a secure channel provided by the shared long-term key K_G.

In the second case, if Dev is unavailable, then the secret r can be recovered by $node_j$ $(1 \leq j \leq N')$ using their shares (j, sh_j) $(1 \leq j \leq N')$. Then the corresponding data collection key and pairwise keys for $node_v$ can be generated and transmitted to $node_v$ in the same way as the above case.

4 Scheme Secure against Aggressive Adversaries

In this section, we propose a key distribution scheme which is secure against aggressive adversaries. Compared with the previous scheme, we use public key cryptographic techniques, including digital signature schemes and Diffie-Hellman key exchange, in order to deter the effect of compromised nodes by aggressive adversaries. Nonetheless, both key distribution schemes make use of the secret sharing technique, therefore, the remarks in Section 3.3 apply to this scheme and we skip it here [1].

[1] The only difference is that a secure channel between Dev and a new node can be provided using a symmetric key resulted from a standard Diffie-hellman key exchange.

4.1 Preliminaries

Digital signature schemes provide a means by which an entity can bind its identity (or public key) to a piece of information (usually referred to as a message). A digital signature scheme is made up of the following algorithms [15]:

1. KeyGen: which takes a security parameter ℓ as input, and outputs a public (verification) key pk and a private (signing) key sk.
2. Sign: which takes as input a message m and a private key sk and produces a signature σ for the message m.
3. Verify: which takes as input a message m, a public key pk and a signature σ, and outputs either accept (denoted by 1) or reject (denoted by 0).

The existential unforgeability of a digital signature scheme is defined as follows:

Definition 4. *A digital signature scheme is existentially unforgeable under an adaptive chosen message attack if the probability of success of any polynomially bounded attacker in the following game is negligible. The attack game is carried out between an attacker $\mathcal{A}$ and the hypothetical challenger $\mathcal{C}$.*

1. *Initialisation: $\mathcal{C}$ runs* $\mathsf{KeyGen}(\ell)$ *to generate a public key pk and a private key sk.*
2. *Challenge: The attacker runs $\mathcal{A}$ on the input pk and terminates by outputting a pair m^*, σ^*. During its execution, $\mathcal{A}$ can query the* Sign *oracle with any input m $(m \neq m^*)$.*

The attacker wins the game if $\mathsf{Verify}(m^*, pk, \sigma^*) = 1$*, and, the attacker's advantage is defined to be* $\Pr[\mathsf{Verify}(m^*, pk, \sigma^*) = 1]$.

Given a group $\mathbb{G}$ of order p, the computational Diffie-Hellman assumption holds if, given g^x and g^y where x, y are randomly chosen from $\mathbb{Z}_p$, an adversary can compute g^{xy} only with a negligible probability.

4.2 Description of the Proposed Scheme

In the *system setup* phase, the trusted device Dev selects a digital signature algorithm (KeyGen, Sign, Verify) and a public/private key pair (PK_G, SK_G). It also chooses a group $\mathbb{G}$ for Diffie-Hellman key exchange [5] and a finite field $\mathbb{F}$ for Shamir's secret sharing.

In the *node setup* phase, every node $node_i$ is initialized by the trusted device Dev: a public/private key pair (PK_i, SK_i) is generated and the parameters $(PK_G, Cert_i, PK_i, SK_i, \mathbb{G}, \mathbb{F})$ are stored in the node, where $Cert_i$ is a signature of $PK_i||ID_i$ signed with SK_G. For simplicity, we assume all nodes have been programmed to perform all the operations in the key distribution scheme. The key distribution scheme is as follows.

1. A node $node_i$, which is connected to the trusted device Dev, becomes a sink node, broadcasts a bootstrap message to the network. The bootstrap message is defined as follows.

$$node_i + \mathsf{Dev} \rightarrow * : ctr,\ g^{r_i},\ \mathsf{Sign}_{SK_G}(ctr||g^{r_i}). \tag{4}$$

2. After receiving the bootstrap message, every node $node_j$ verifies the signature. If the signature is not valid or the value of ctr is smaller than the local counter value, $node_j$ terminates by broadcasting a failure message. Otherwise, it sets its local counter value to be ctr, and sends the following message to the sink node.

$$node_j \rightarrow node_i : g^{r_j}, \mathsf{Sign}_{SK_j}(ctr||g^{r_i}||g^{r_j}). \tag{5}$$

The node $node_j$ computes two ephemeral keys ek_j and ek'_j, where

$$ek_j = \mathsf{H}(1||g^{r_i r_j}||ctr||ID_i||ID_j),\ ek'_j = \mathsf{H}(2||g^{r_i r_j}||ctr||ID_i||ID_j).$$

3. After receiving the message from $node_j$, the sink node first checks the counter value and the signature. If the check fails, it terminates by broadcasting a failure message. Otherwise, it continues. *At a certain point, the sink node learns that session keys need to be distributed to a group of nodes, say* $node_j$ $(1 \leq j \leq N')$ *and* $N' \leq N$. The sink node computes an ephemeral key pool $\Gamma = \{ek_1, ek_2, \cdots, ek_{N'}, ek'_1, ek'_2, \cdots, ek'_{N'}\}$, where for $1 \leq j \leq N', j \neq i$

$$ek_j = \mathsf{H}(1||g^{r_i r_j}||ctr||ID_i||ID_j),\ ek'_j = \mathsf{H}(2||g^{r_i r_j}||ctr||ID_i||ID_j).$$

The sink node then does the following.

(a) Using Shamir's (t, N)-threshold secret sharing technique, generate N shares $\{(j, sh_j)|1 \leq j \leq N\}$ to hide a secret $r \in_R \mathbb{F}$.

(b) Send the following message to the node $node_j$

$$\begin{aligned} node_i \rightarrow node_j : &\mathsf{ENC}_{ek_j}(ctr||j||sh_j||sk_j||T_j), \\ &\mathsf{MAC}_{ek'_j}(ctr||\mathsf{ENC}_{ek_j}(ctr||sk_j||T_j)), \end{aligned} \tag{6}$$

where $sk_j = \mathsf{H}(3||ctr||ID_j||r)$ and T_j is a concatenation of $pk_{t,j}$ for all $ek_t \in \Gamma$ and $t \neq i$. The value $pk_{t,j}$ is set to be $\mathsf{H}(4||ctr||ID_t||ID_j||r)$ if $t < j$, and $\mathsf{H}(4||ctr||ID_j||ID_t||r)$ otherwise. Consequently, $pk_{t,j} = pk_{j,t}$ holds.

4. After receiving the message, $node_j$ first checks the MAC code. If the check fails, it terminates by broadcasting a failure message. Otherwise, it decrypts $\mathsf{ENC}_{ek_j}(j||sh_j||sk_j||T_j)$ to obtain the data collection key sk_j, pairwise keys T_j, and the share (j, sh_j). It also update ctr to be $ctr + 1$.

Lemma 2. *The proposed scheme is secure under Definition 2 based on the computational Diffie-Hellman (CDH) assumption, given that the digital signature scheme is existentially unforgeable, the encryption algorithm is a pseudorandom function, and* H *is a random oracle.*

Proof sketch. Suppose that an adversary has the advantage ϵ the attack game shown in Fig. 5. We first have the following observation.

Observation. *During the attack game, in the session identified by ctr^*, $node_j$, for any j such that $node_j$ is involved in the session, is supposed to receive the following value:*

$$ctr^*,\ g^{r_i},\ \mathsf{Sign}_{SK_G}(ctr^*||g^{r_i}),$$
$$\mathsf{ENC}_{ek_j}(ctr^*||j||sh_j||sk_j||T_j),\ \mathsf{MAC}_{ek'_j}(ctr^*||\mathsf{ENC}_{ek_j}(ctr^*||sk_j||T_j)).$$

Based on the existential unforgeability of the signature scheme, the probability that the first message is not generated (or, simulated) by the challenger is negligible. Based on the CDH assumption and the existential unforgeability of the MAC algorithm, the probability that an adversary can forge the second message is negligible given that H *is a random oracle. Therefore, these values are generated by the challenger, and the proof is straightforward so that we skip it here.*

The rest of the security proof is done through a sequence of games [18].

Game_0: In this game, the challenger faithfully simulates the protocol execution and answers the oracle queries from $\mathcal{A}$. Let $\delta_0 = \Pr[b' = b]$, as we assumed at the beginning, $|\delta_0 - \frac{1}{2}| = \epsilon$.

Game_1: The challenger performs faithfully as in Game_0, except that the challenger stops if the values described in the above observation are not generated by the challenger (referred to as the event Ent_1). Let $\delta_1 = \Pr[b' = b]$ at the end of this game. From the Difference Lemma in [18], we have $|\delta_1 - \delta_0| \leq \Pr[Ent_1]$ which is negligible.

Game_2: The challenger performs faithfully as in Game_1, except that, in the session identified by ctr^*, in step 3 of the scheme the messages sent to $node_j$, for any j such that $node_j$ is involved in the session and $node_j$ has not been issued any $\mathsf{Corrupt}_{\mathsf{I}}$ query, are replaced with the following, where Ran_j is a random function.

$$\mathsf{Ran}_j(j||sh_j||sk_j||T_j),$$
$$\mathsf{MAC}_{ek'_j}(ID_j||ctr^*||\mathsf{Ran}_j(j||sh_j||sk_j||T_j)),$$

Since H is a random oracle and the encryption algorithm is a pseudorandom function, Game_2 is identical to Game_1 unless the event Ent_2 occurs: the adversary has queried H with $*||r||*$ or $*||g^{r_i r_j}||*$ for any j such that $node_j$ has not been issued any $\mathsf{Corrupt}_{\mathsf{I}}$ query. Based on the CDH assumption and the security of the Shamir secret sharing scheme, $\Pr[Ent_2]$ is negligible. Let $\delta_2 = \Pr[b' = b]$ at the end of this game. From the Difference Lemma in [18], we have $|\delta_2 - \delta_1| \leq \Pr[Ent_2]$ which is negligible.

In Game_2, since the encryption is provided by random functions, the probability $\delta_2 = \frac{1}{2}$. As a result, we have

$$\begin{aligned}\epsilon &= |\delta_0 - \frac{1}{2}| \\ &\leq |\delta_1 - \delta_0| + |\delta_2 - \delta_1| + |\delta_2 - \frac{1}{2}| \\ &\leq \Pr[Ent_1] + \Pr[Ent_2]\end{aligned}$$

Since $\Pr[Ent_1]$ and $\Pr[Ent_2]$ are negligible, the lemma now follows. □

5 Performance Analysis

Based on the theoretical results, here we give a performance analysis of KALwEN+. Let T_e be the time for a symmetric key encryption, and T_m be the time for computing a MAC value. Let T_p be time for one exponentiation computation. T_s denotes the time for the (t, N)-threshold secret sharing algorithm which is used in KALwEN+. Let T_{sig} and T_{ver} be the time costs for generating and verifying a signature, respectively. For a gossip sensor network with n nodes, the performance of KALwEN+ is estimated as follows.

Table 1. The Performance Estimation of KALwEN+

KALwEN+	Against Active Adversary	Against Aggressive Adversary
Sink node costs	$(n+1)T_e+(n+1)T_m+1T_s$	$1T_{sig}+(n+1)T_p+nT_e+nT_m+1T_s$
Member node costs	$2T_m+2T_e$	$1T_{ver}+1T_{sig}+1T_p+1T_e+1T_m$
Communication rounds	3-Rounds	3-Rounds
Storage costs	O(n)	O(n)

For the estimated performance, the potential bottleneck of the scheme will be the sink node. Especially in a large network, a typical sensor node can hardly afford the computational costs of (t, N)-threshold secret sharing by itself. Since the sink node can be connected to a trusted device, the computational costs would possibly be shared by the device while the scalability of network is large.

6 Conclusion

By simply using the Shamir's secret sharing techniques and the Diffie-Hellman algorithm, a family of novel key management schemes that named KALwEN+ has been proposed for wireless medical sensor network. The KALwEN+ schemes can be fully based on broadcast communication, and does not require special equipment like some existing schemes do. The secret sharing technique used in KALwEN+ not only supports efficient node addition/removal, but also elegantly ensures security against key-exposure. For applications with highly-constrained resources, the KALwEN+ scheme that fully based on symmetric cryptographic primitives is a reasonable choice. For future work, we will investigate the practical performance and the interoperability of KALwEN+ in a multi-user scenario.

Acknowledgement. We would like to thank Frits van der Wateren and Teun Hendriks for their helpful advice during ALwEN workshops. And also thank many anonymous reviewers for their valuable comments. Zheng Gong acknowledges the support of SenterNovem for the ALwEN project, grant PNE07007. Yee Wei Law is supported by the Australian Research Council Research Network on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), and the ARC DP1095452.

References

1. AAL. European union. the ambient assisted living (aal) joint programme (January 2008), http://www.aal-europe.eu/about-aal
2. ALwEN. Ambient living with embedded networks (January 2010), http://www.alwen.nl
3. Anderson, R.: A security policy model for clinical information systems. In: IEEE Symposium on Security and Privacy, pp. 30–43 (1996)
4. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
5. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory IT-22(6), 644–654 (1976)
6. Dutta, R., Chang, E.-C., Mukhopadhyay, S.: Efficient self-healing key distribution with revocation for wireless sensor networks using one way key chains. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 385–400. Springer, Heidelberg (2007)
7. Dutta, R., Mukhopadhyay, S., Dowling, T.: Generalized self-healing key distribution in wireless adhoc networks with trade-offs in user's pre-arranged life cycle and collusion resistance. In: Q2SWinet 2009: Proceedings of the 5th ACM Symposium on QoS and Security for Wireless and Mobile Networks, pp. 80–87. ACM Press, New York (2009)
8. Eschenauer, L., Gligor, V.D.: A key-management scheme for distributed sensor networks. In: CCS 2002: Proceedings of the 9th ACM Conference on Computer and Communications Security, pp. 41–47. ACM, New York (2002)
9. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM 33(4), 792–807 (1986)
10. Haas, Z.J., Halpern, J.Y., Li, L.: Gossip-based ad hoc routing. IEEE/ACM Transactions on Networking (TON) 14(3), 479–491 (2006)
11. Halperin, D., Heydt-Benjamin, T.S., Ransford, B., Clark, S.S., Defend, B., Morgan, W., Fu, K., Kohno, T., Maisel, W.H.: Pacemakers and implantable cardiac defibrillators: Software radio attacks and Zero-Power defenses. In: 29th IEEE Symposium on Security and Privacy, Oakland, California, pp. 129–142. IEEE Computer Society, Los Alamitos (2008)
12. Kausar, F., Hussain, S., Park, J.H., Masood, A.: Secure group communication with self-healing and rekeying in wireless sensor networks. In: Zhang, H., Olariu, S., Cao, J., Johnson, D.B. (eds.) MSN 2007. LNCS, vol. 4864, pp. 737–748. Springer, Heidelberg (2007)
13. Khalili, A., Katz, J., Arbaugh, W.A.: Toward secure key distribution in truly ad-hoc networks. In: IEEE/IPSJ International Symposium on Applications and the Internet Workshops, p. 342 (2003)
14. Law, Y., Moniava, G., Gong, Z., Hartel, P., Palaniswami, M.: KALwEN: A New Practical and Interoperable Key Management Scheme for Body Sensor Networks. In: Security and Communication Networks (2010) (in press)
15. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)
16. Oliveira, L.B., Wong, H.C., Bern, M., Dahab, R., Loureiro, A.A.F.: Secleach - a random key distribution solution for securing clustered sensor networks. In: IEEE International Symposium on Network Computing and Applications, pp. 145–154 (2006)
17. Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (1979)
18. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs (2006), http://shoup.net/papers/
19. Yang, G.Z.: Body Sensor Network. Springer, London (2003)