Scalable security in wireless sensor and actuator networks (WSANs): Integration re-keying with routing

Fei Hu
Waqaas Siddiqui
Krishana Sankar

Follow this and additional works at: http://scholarworks.rit.edu/article

Recommended Citation

This Article is brought to you for free and open access by RIT Scholar Works. It has been accepted for inclusion in Articles by an authorized administrator of RIT Scholar Works. For more information, please contact rit.scholarworks@rit.edu.
Scalable security in Wireless Sensor and Actuator Networks (WSANs): Integration re-keying with routing

Fei Hu a,*, Waqaas Siddiqui a, Krishna Sankar b

a Department of Computer Engineering, RIT, 83 Lomb Memorial Drive, Rochester, NY 14623-5603, United States
b CISCO System Inc., USA

Received 2 November 2005; received in revised form 20 April 2006; accepted 28 April 2006
Available online 9 June 2006

Abstract

Our research aims to address the challenging security issues in Wireless Sensor and Actuator Networks (WSANs), a special type of Wireless Sensor Networks (WSNs). Since WSANs have specific network constraints and data transmission requirements compared to general ad hoc networks and other wireless/wired networks, we propose to seamlessly integrate WASN security with a ripple-zone (RZ)-based routing architecture that is scalable and energy-efficient. In this research, we will also develop a two-level re-keying/re-routing schemes that cannot only adapt to a dynamic network topology but also securely update keys for each data transmission session. Moreover, to provide the security for the in-networking processing such as data aggregation in WSANs, we define a multiple-key management scheme in conjunction with our proposed Ripple-Zone routing architecture. Extensive simulations and hardware experiments have been conducted to verify the energy-efficiency and security performance of our security scheme for WSANs.

Keywords: Wireless Security; Wireless Actuator and Sensor Networks; Ad hoc Networks; Cluster-based routing protocol

1. Introduction

1.1. Wireless Sensor and Actuator Network Security

More recently, an important type of network, which is based on the integration of Mobile Ad hoc Network (MANET) that consists of mobile “actuators” and Wireless Sensor Networks (WSNs) with large amount of low-energy tiny “sensors”, have played more and more important roles in homeland security applications [1,2]. Such hybrid networks are usually called Wireless Sensor and Actuator Networks (WSANs) [2], which cannot be simply regarded as MANETs due to the coexistence of mobile actuators (forming MANET) and fixed sensors (forming WSN). As shown in Fig. 1, actuators execute corresponding tasks based on the collected sensing data from sensors. Please notice that there are some important differences between
the two components in a WSAN (i.e., the MANET and the WSN): (1) the number of the nodes in a WSN is significantly larger than in a MANET; (2) sensors are usually low-cost devices with severe constraints with respect to energy source, computation capabilities, and memory; while actuators generally have relatively higher energy storage, which allows longer wireless transmission distance; (3) the sensors are usually stationary or with quite limited mobility; and (4) mode of communication in WSNs typically is many-to-one (from sensors to sink), while it is typically peer-to-peer in MANETs. While WSNs are concerned only with sensor-to-sink interconnections, in WSANs four types of coordination need to be considered in the same scenario: actuator-to-actuator (A–A) (to determine which actuators should respond to which sensing area), sensor-to-sensor (S–S) (to use multi-hop communication mode to transmit sensing data), actuator-to-sensor (A–S) (downlink transmission to instruct sensors to execute a certain sensing tasks), and sensor-to-actuator (S–A) (uplink transmission to report new events or required query results). A–A coordination can be regarded as a MANET issue that has been studied extensively so far. S–S coordination is a topic of WSN that is still a largely unexplored field.

Security is important in many WSAN-based civilian/military applications, such as disaster recovery (earthquake and fire rescue, etc.), airport terrorist-attack prevention, industrial manufacturing control, and so on. The study of security should consider the following challenges in WSANs: (1) considering actuator/sensor coordination and actuator/sensor heterogeneity; (2) protocol simplicity (sensors have limited memory and computational capability [3]); (3) algorithm scalability (there can be hundreds, if not thousands, of sensors in a typical WSAN application [1]); and (4) low-energy consumption (to extend the lifetime of tiny sensors). The energy for one bit of wireless communication can be used to execute over 1100 local instructions in a sensor [2]. Thus the WSAN trustworthiness protocols should have low wireless communication overhead.

Many of the current sensor network security schemes are based on key pre-distribution strategy [6–8]. It works as follows: before sensor deploy-
ment, a subset of key pool are assigned to each sensor to make two sensors likely share a pairwise key. However, we argue that key pre-distribution only cannot achieve satisfactory security performance because the attackers can capture some sensors/actuators and learn those “permanent” keys. It is important to update those keys (i.e., using re-keying) from time to time for different packet transmission sessions. Re-keying is also important in terms of adaptation to network topology changes due to node failure, node addition, and interruptions in the wireless transmission medium [5]. For example, if nodes fail due to low power, messages will fail to be delivered because routes containing the dead nodes still exist. In the case of node addition, it is important to distinguish between legitimate sensor traffic, and the infiltration of the network by an enemy node. Also, intermittent connectivity must be considered because the security scheme should be able to deal with wireless errors. These considerations, along with the resource (battery, memory, etc.) constraints imposed, make the design of a WSAN security scheme an extremely difficult task.

The contributions and innovations of our proposed WSAN security scheme include the following four aspects:

1) Seamless integration of security with scalable WSAN routing protocols: our scheme is highly practical because it was designed to integrate routing layer and security protocol without sacrificing power. It is a dynamic, distributed protocol where security is provided independent of central control. Existing work overlooks the idea that any security scheme should be seamlessly integrated with the special characteristics of sensor network architecture, especially routing protocols; otherwise, the security scheme may not be practical or energy-efficient from the network protocol point of view [3]. Our security considers special WSAN topology through a two-level keying.

Thorough research of the field has found that most of the existing sensor network security strategies focus only on key management/security algorithms. For example, all existing key-predistribution schemes try to establish pairwise keys between each pair of nodes. However, most sensors do not need to establish a direct point-to-point secure channel with sensors multiple hops away since sensor network use hop-to-hop communication techniques to achieve long-distance transmission. One of the most famous schemes, SPINS [9], simply assumes a flooding-based, spanning-tree architecture with the BS as the tree-root. However, the establishment and maintenance of a global spanning tree in a large-scale WSN with a large footprint may not only bring unacceptable communication overhead (and thus increased energy consumption1), but also cause a large transmission delay and make the assumption of time synchronization in µTESLA (a broadcast authentication protocol [9]) impractical. Another important feature of our work is that it has a robust hop-to-hop transmission scheme and can recover from multiple-key losses.

2) Dynamic security: dynamic network topology is native to WSANs because nodes can fail or be added. In the case where nodes fall out, these nodes

---

1 Most of the sensor’s power is consumed by communication, not instruction processing [3]. The energy used to communicate one bit of data can be used to execute over 1000 local instructions [3]. High-energy consumption can drain sensors quickly and thus shorten the network lifetime.
must be recognized “dysfunctional” from the viewpoint of the rest of the network. In the case of node addition, a protocol must be able to distinguish between legitimate node addition and attempted enemy infiltration. Given these reasons, “adaptive security” should be present for WSAN applications in order to ensure overall network security.

(3) Robust re-keying: from time to time, network enemies might compromise sensors and all security information in those sensors may be obtained. Therefore, after key-predistribution and sensor deployment, a re-keying scheme should be used to update some types of keys such as group keys (for broadcast security) and session keys (for securing the current data packets). This is done to ensure that enemies cannot acquire the keys easily. In this work, a re-keying protocol that can adapt to dynamic network conditions such as sensor compromise and topology change is planned.

(4) Low-complex implementation. Our work uses a symmetric-key-based scheme because memory use is a major concern in sensors [3]. This prohibits the use of memory-intensive asymmetric keying schemes. Asymmetric keying schemes need more complex cryptography calculations and protocols, which can bring too much communication overhead compared to symmetric-key-based schemes. Our security protocol also has low transmission energy due to its cluster-based key management. Because a WSAN typically consists of hundreds, if not thousands, of nodes, network topology/densities can change; therefore, a centralized or flooding-based security scheme cannot scale well. Thus distributed algorithms and localized coordination to achieve global convergence and scalability is preferred [3].

The rest of this paper is organized as follows: in Section 2, we will point out the shortcomings of the related works and the importance of this work; Section 3 first provides our scalable routing architecture and then discusses about the security issues in high-level nodes of WSAN (i.e., among actuators); in Section 4, security among low-level nodes (among sensors in each actuator’s domain) will be described in detail; extensive simulation results will be given in Section 5. Section 6 is the security analysis; hardware experiments are stated in Section 7. Finally, Section 8 concludes this paper.

1.2. Related works

As pointed out in [2], currently, very little research work has been conducted on WSANs that have the coexistence of mobile actuators and low-energy sensors. A contention-free MAC protocol for WSAN is presented in [24]. In [25], WSANs are only examined from the control engineering perspective. While the A–A coordination is investigated in [26], existing and emerging technologies in WSANs are briefly described without consideration of the interaction between the sensors and actuators in [27,28].

The closest related work to WSANs occurs in the general Wireless Sensor Networks (WSNs), which is a hot research field today. A survey of the early work on WSNs is provided in [3]. In terms of security issues in general sensor networks (not WSANs), the pioneering work on securing WSN end-to-end transmission is SPINS [9], which requires time synchronization among sensors. It also proposes μTESLA, an important innovation for achieving broadcast authentication of any messages sent from the base station (BS). An improved multi-level μTESLA key chain mechanism was proposed in [29,30]. A key-pool scheme was suggested in [10] to guarantee that any two sensors share at least one pairwise key with a certain probability. Multiple pairwise keys may be found between nodes by the schemes proposed in [11]. Key-predistribution schemes utilizing location information were described in [12]. Other WSN security research works include authentication [13], Denial-of-Service (DoS) attacks [14], routing security [15], group security [16,17], multiple-key management [18,19], and simple system-level security analysis [20–23].

Why those WSN “security” schemes do NOT work well in WSAN environments? One of the common drawbacks of those sensor network security schemes is that they do not integrate security with a hierarchical sensor network routing architecture and specific characteristics of WSANs (such as the coexistence of actuators and sensors). Because the sensors may only want to report data to the nearby actuators, it will cause much security overhead if we build secure links between any two nodes. It is necessary to reduce key management overhead through cluster-based communication architecture around each actuator [2,3]. In this research, we integrate WSAN security issues with our proposed ripple-zone-based WSAN routing architecture (Sections 3 and 4). We will show that clustering-based re-keying scheme can save lots of energy compared to those works based on general flat routing topology. (Note that energy consumption is the top concern in tiny, battery-driven sensors [3]).
2. Routing assumptions

In this research, we mainly compare three types of routing schemes that serve the basis of keying protocols:

2.1. Flat-based routing

As shown in Fig. 2, most of traditional sensor network security schemes [9,12] just focus on end-to-end security issues (top diagram) and ignore WSAN routing details. They do not build security above the low-energy routing architecture and just simply assume the entire network uses tree- or flat-based topology. Without concern on the routing details, it is impossible to prevent person-in-middle attack or hop-to-hop security. Our scheme integrates security with WSAN topology discovery procedure (Fig. 2 bottom).

2.2. General clustering-based routing

A related work close to ours is Pebblenets (Fig. 3) [31]. It proposes a cluster-based routing architecture. However, it has several major shortcomings: (1) it only assumes a simple homogeneous architecture (i.e., only sensors) and thus does not apply in WSAN platforms that have both high-energy actuators and low-energy sensors in one network. Our scheme will use a two-level routing/security protocol
to achieve key refreshing among high-level actuators and low-level sensors; (2) another downfall to Pebblenets is the idea of having all keys being generated by a backbone sensor node. Therefore, if a crucial backbone sensor is captured during key generation, new keys can be known, resulting in full system compromise. In our scheme, the BS and actuators together control new key generations. We use the concepts of Backbone Key and Session Key (Section 3.1) to securely update keying information; (3) we have also made other significant improvements such as the concept of “ripple key” that can achieve asynchronous broadcast authentication (i.e., without synchronization assumption as in μTESLA [9]). (See Section 4 for details.)

2.3. Our proposed Ripple-zone-based routing

As the prerequisite of WSAN trustworthiness, we argue that it is very important to design a hierarchical, energy-efficient routing scheme compatible with the specific network characteristics of WSANs since both reliable packet transmission (internal trustworthiness) and security key management (external trustworthiness) need a low-energy routing protocol. We have created a Member Recognition Protocol (MRP) to allow actuators and sensors to self-organize themselves into separate “domains” with each actuator as the domain center (please see the author’s paper [32,33] for details on WSAN routing establishment scheme). After running our MRP, each actuator will be aware of its domain members.

As shown in Fig. 4, within the domain of each actuator, we further propose the concept of a Ripple-Zone (RZ) around each actuator, in which sensors are assigned to different “ripples” based on their distances, in number of hops, from their actuator. Some sensors are then chosen as “masters”, or called “Cluster-Heads” (CHs), based on our Topology Discovery Algorithm (TDA) [32]. Each “master” aggregates data from the sensors in its zone before it transmits data to a “master” in next “ripple” that is closer to the actuator, i.e., with a smaller number of hops to the actuator.

In the rest of this paper, when we mention “node”, it can refer to an actuator, a CH, or a common sensor in a zone. We use CHs/Masters, Sink/Base station (BS) alternatively. In Section 4.5, we will further discuss the secure routing establishment procedure among sensors.

Due to sensor failure/addition, the above RZ-based routing architecture may break down. Thus our above routing algorithm needs to run periodically. Obviously, the faster the topology changes, the shorter the routing update interval. Here let us determine the proper update interval value, T. We denote d as the sensor radio range. Suppose the total number of sensors is n and p is the probability of a sensor becoming CH. Because at any time there are np zones, the average number of sensors in each zone, n’, should be
\[ n’ = \frac{n}{np} = \frac{1}{p}. \]
We denote pin as the probability that a current sensor fails in its zone, and pout as the probability that a new sensor is added the current zone. Thus, the probability that no new sensor enters the current zone is:
\[ p_{no-in} = \frac{1}{np} \]
and the probability that no sensor fails in the current zone is
\[ p_{no-out} = (1 - p_{in})^{n’}. \]
The probability that the zone changes topology is then as follows:
\[ P_{update} = 1 - p_{no-in} p_{no-out} \]
\[ = 1 - (1 - p_{out})^{n-1/p} (1 - p_{in})^{1/p} \]  (1)

![Fig. 4. Proposed ripple-zone-based WSAN routing.](image)
The value of $p_{in}$ and $p_{out}$ can be determined by the conclusions from [37] and is approved to be the function of sensor amount changing speed and $d$. Given a target ‘topology change probability’, we can determine the topology update interval such that the practical topology change probability is below that target value. Suppose 500 sensors are randomly distributed in an area with a radius of 1 km and each sensor has a transmission range of 20 m. Also suppose each zone has an average 50 sensors, with average sensor amount changing speed as five sensors/s. If we want to make the topology change probability less than 10% (of the target topology change probability), we get the tree-zone updating interval of 5.5 s, Fig. 5 shows the relationship of routing update interval $\Delta$ as a function of topology change speed.

3. High-level security

Security assumptions: before the discussion of our security scheme, we make the following reasonable assumptions just as in other sensor network security schemes [10,11,29,30]:

1. The BS is always trustworthy and is located in a safe place without power or memory limitation.
2. Before node deployment (i.e., in the pre-deployment phase), all the sensors/actuators share an initial global key with the base station (BS).
3. Each of the sensor or actuator also shares a 1:1 initial pairwise key with the BS.
4. Each sensor has a unique ID with enough length (just like an unique credit card number). Thus the BS can use sensor ID to distinguish between legitimate IDs and illegitimate ones.
5. All the above keying or ID information is stored in a semi-permanent memory and will become invalid or be automatically self-erased after a pre-determined expiration timer. For instance, if the network deployment time is 11 AM, we can set up the expiration timer to 11:05 AM (i.e., 5 min of valid time). We assume that the enemies need a certain time (at least a few minutes) to break the sensor and get to know the security information. Immediately after the sensors are dropped to some area, those sensors will use the initial global key, initial pairwise key and sensor ID to securely establish the above ripple-zone-based WSAN routing architecture. Once the routing architecture is established and recognized in all the actuators, sensors, CHs, re-keying protocol will be used to periodically refresh all types of keys (see Section 3.1 and Section 5).

When a new node joins in a WSAN, it will also use the above keying information (i.e., initial global key, initial pairwise key and sensor ID) to get recognized by the existing nodes.

3.1. High-level security (among actuators)

Based on the above RZ-based routing architecture, we further propose a low-energy, low-complexity, and hierarchical (two-level-based) key management scheme.

In the High-Level² (among actuators), two types of keys exist: (1) a Session-Key (SK) is used to secure the transmission of data packets. (2) A Backbone Key (BK) is used to secure control packets that include SK re-keying information. Fig. 6 shows the relationship between these two keys. SKs need to be re-keyed periodically to defeat attacks. However, the BK is refreshed in an event-triggered way (typical events include actuator insertion, death, or compromise). The WSAN base station (i.e., the sink) can use any well-known Group Communication Security protocol [34,35] to perform BK re-keying among actuators.

To ensure secure SK re-keying, the sink initially use BK management schemes [34,35] to encrypt, authenticate and transmit a control packet, which includes a SK to be used in the current sensing data transmission. Note that Control packets are different from data packets. The former is secured by BK management schemes while the latter should be secured by SKs.

Every a certain interval, a new SK will be distributed to all actuators. All new SKs are derived from a one-way hash function $H$ as in [9]. We do not use a

---

2 Next Section (i.e., Section 4) will discuss the “low-level security” among sensors belonging to the domain of an actuator.
MAC (Message Authenticated Code) for SK authentication since receivers can use the one-way property of SK sequence to check whether the received SK belongs to the same sequence or not. We call such a SK authentication concept as implicit authentication, which works based on the following key chain scheme (Fig. 7), which is also used in the LiSP scheme [36]:

The sink first pre-computes a long one-way sequence of keys: \( \{SK_M, SK_{M-1}, \ldots, SK_n, SK_{n-1}, \ldots, SK_0\} \) (size \( M \gg n \)), where \( SK_i = H(SK_{i+1}) \). Initially only \( SK_n \) (instead of the whole \( M \)-size key sequence) is distributed to each actuator. Then an actuator can utilize \( H \) to figure out \( SK_{n-1}, \ldots, SK_0 \). The \( n \) keys \( \{SK_n, SK_{n-1}, \ldots, SK_1\} \) are stored in a local key buffer with a length of \( n \). However, \( SK_0 \) is not in the buffer because it is used for the current data packet encryption/decryption. After the initial \( SK_n \) delivery, the sink periodically sends \( SK_{n+1}, \ldots, SK_M \) (one key distribution in each period) to all actuators. To achieve communication confidentiality, each new key is encapsulated into a re-keying control packet that is encrypted with the BK or the currently active SK.

After receiving a new SK, the actuator keeps applying \( H \) to it for some times, trying to find a key match in its key buffer. For instance, assume that an actuator receives a new key \( SK_j \). Also assume that its key buffer already holds \( n \) SKs as follows: \( \{SK_i, SK_{i-1}, \ldots, SK_i - n + 1\} \). If the actua-
tor finds out that $H(H(H(H(SK_i)))) \notin \{ SK_i, SK_{i-1}, \ldots, SK_{i-n+1} \}$, i.e., implicit authentication fails, SKj will be discarded. Otherwise, if implicit authentication is successful, the key buffer is shifted one position. The SK shifted out of the buffer is pushed into the “active key slot” to be used as the current SK (see Fig. 7), and the empty position is filled in with a new key $SK'$. $SK'$ is derived from the received SKj through $H$ and meets the following two conditions:

$$SK' = H(H(H(H(SK_j)))) \quad \text{and} \quad H(SK') = SK_i.$$  

The reason for using a key buffer is to tolerate multiple-key losses. As shown in Fig. 8, even if up to $(n-1)$ SKs are lost due to unreliable wireless channels (such as radio shading, fading and communication noise), as long as the nth SK received is authenticated, we can still restore the lost SKs in the key buffer by applying the one-way hash function. By using a key buffer, the lost SK will not influence the current security session until a maximum delay of $(n-1)$ re-keying periods. This feature also makes SK management robust to clock skews.

As discussed above, SK re-keying is conducted periodically to overcome active attacks (Fig. 5). However, the SK should be reset immediately under any of the following conditions:

- one of the actuators is compromised by attackers;
- an actuator runs out of energy and thus stops operations;
- a new actuator is added;
- an actuator loses $n+1$ keys and thus cannot deduce any lost keys;
- the sink distributes all M keys and needs to build a new key sequence.

If the sink detects topology change (due to actuator addition/death) or detects node compromise through intrusion detection schemes, it immediately renews the backbone key to $BK_{new}$ through group communication protocols [34] and generates a new key sequence $\{ SK'_M, SK'_{M-1}, \ldots, SK'_n, SK'_{n-1}, \ldots, SK'_0 \}$ ($M \gg n'$), $SK'_j = H(SK'_{j+1})$. Where $n'$ is new key buffer size in each actuator.  

The sink then sends out a new group of parameters to each actuator as follows:

$$Sink \rightarrow actuator : E_{BK_{new}}(A'[n'|SK'_{n'}])$$  

(2)

where $A'$ is the new re-keying period.

On receiving these new parameters, an actuator will perform the following tasks: (1) delete all old SKs in the previous buffer; (2) rebuild a new buffer with length of $n'$; (3) calculate the remaining $n'$ keys $\{ SK'_{n'-1}, \ldots, SK'_1, SK'_0 \}$ based on $SK'_n$ and the hash function $h$; (4) Choose $SK'_0$ as the current active session key for data encryption; and (5) invoke a new timer that expires every $A'$. When timer expires, the sink sends out a new key based on the following rule:

$$Sink \rightarrow Actuator : E_{BK}(SK'_{n'+i+1}), \quad \text{at time } t = iA'$$  

(3)

If the sink has distributed the whole key sequence (after $n \times A'$), it will also send out a new set of parameters as shown in (2). If an actuator misses $n$ keys, the sink must send a message with $E_{BK}(A'[n'|SK'_C])$ to that actuator, where $SK'_C$ is the current Session Key.

We can achieve better security by using a shorter re-keying period $A'$ (i.e., more frequent key updating). However, this will lead to greater communication overhead and increased hash calculation. There also exist other tradeoffs. For instance, the larger the buffer size $n'$, the more tolerant to key loss an actuator is, which is because it can recover more

---

3 Each time we require the actuators to create buffers with a new size $n'$ to make attacks more difficult.
missed keys based on one-way hash function. However, a large buffer requires more memory and calculation overhead.

Fig. 9 shows the procedure of the security protocol in the high-level nodes (actuators). It is implemented in our simulations (Section 5) and experiments (Section 7).

4. low-level security

4.1. Low-level (in the domain of each actuator) re-keying

A unique issue in WSAN security is that the selection of key sharing schemes should consider the impact on in-networking processing [9]. For example, data aggregation is necessary for reducing communication overhead from redundant sensed data. If one simply adopts a key scheme like pairwise key (that is shared between only two nodes), memory limitations will prohibit a ‘master’ from maintaining all the keys necessary to aggregate data from its zone sensors. In addition, building an end-to-end secure channel between any two nodes is inadvisable, because intermediate sensors/actuators may need to decrypt and authenticate the data collected from multiple sensors. Since different types of messages exchanged among sensors have different security requirements (such as data aggregation security), a single keying mechanism may not be suitable for all cases. Thus, multiple keys need to be introduced in low-level sensors in an actuator domain. We have defined multiple types of keys for different security purposes as follows:

1. Master-to-Actuator Key (MAK): an MAK is shared between each master and its Domain Actuator. It is used for direct Master-to-Actuator secure communication. An MAK is generated based on a high-Level Session Key (SK) as follows: $MAK = f_{SK}(Master-ID)$;
2. Inter-Master Pairwise key (MPK): occasionally secure channels need to be established between two masters that belong to two actuator domains;
3. Sensor-to-master Pairwise Key (SPK): a sensor-to-master pairwise key is shared between a master and each of the sensors in its zone;
4. Zone Key (ZK), also called Cluster Key (CK): zone keys are used for data aggregation and also for the propagation of a query message to the whole cluster. Each ZK is shared among all sensors in the same cluster;
5. Ripple Key (RK): a ripple key is used to achieve hop-to-hop security in an actuator domain. To save security overhead, our scheme generates a new key based on a family of Pseudo Random Functions (PRF) $\{f\}$ as follows: $K'_{k,x} = f_{k}(x)$, ($K$ is last key and $x$ is a random number). Section 4.4 provides detail procedure of using Ripple Keys.

4.2. Robustness to key losses

In high level, we use the one-way property of hash functions to recover the lost keys between
two re-keying events. In low level, since we do not use a one-way key chain (because it is not suitable for sensors that have less memory and computational capability than actuators), we recover lost control packets that include key information through ripple-to-ripple retransmission scheme and a local path repair routing protocol [32]. A simple example demonstrating our key loss recovery scheme is shown in Fig. 10, after master B sends out a control packet to A (the closest master to B), B sets up a timer T1. If T1 expires and B has still not received a positive acknowledgement packet from A, B will retransmit the packet to A. If B attempts more than three retransmissions without success, it regards A as unreachable (possibly due to an unreliable wireless link), and our scheme then finds another good path as follows: B checks the reachability of A’s neighboring masters H and E. If either of them is reachable, B will transmit the packet to one of them. If both H and E are unreachable, our protocol goes back one ripple, i.e., to master K. K discovers B’s neighboring masters and tries to deliver the packet. Eventually a new ripple-to-ripple good path will be discovered. We will verify its efficiency through simulation experiments in next section.

4.3. Relationship between SPKs and ZKs

ZKs are very important for securing data aggregation in a local sensing area since they enable the whole zone to encrypt and decrypt data packets without the use of multiple keys. Note that ZK re-keying occurs after SPK re-keying. The basic procedure is as follows: whenever a master M re-generates a new random key \( ZK' = f_{ZK}(x) \) (where ZK is the last Zone Key and x is a random number), it sends \( ZK' \) to each of its zone members using the corresponding SPKs as follows:

\[
M \rightarrow \text{actuator } \Theta_i : \text{Rekeying} \#|\text{Master} ID|E_{SPK(i)} \left( ZK'|\text{Rekeying}\#|\text{Master} ID \right) \tag{4}
\]

Each zone member analyzes and decrypts this message, then stores the new \( ZK' \) in its buffer. The re-keying of ZK is performed periodically (the control timer is the same as Level 1 \( \Lambda \), see Section 3.1).

4.4. Application of our Ripple Key scheme

The aforementioned ripple key scheme can overcome the time synchronization problem [9] when used for Broadcast Authentication, i.e., each zone ensures that a broadcasting message does come from its own Actuator versus other sources. To disseminate a message, an actuator broadcasts a separate message (called \( \text{Cmd}_\text{MSG} \)) for different ripples as follows:

\[
\text{Actuator} \rightarrow \text{Ripple } [i] : \text{Message}|\text{Actuator} ID|\text{Tran} ID|MAC_{\text{RK}(i)}(\text{Message}|\text{Actuator} ID|\text{Tran} ID) \tag{5}
\]

where \( \text{Actuator-ID} \) helps each master distinguish its own actuator from others. \( \text{Tran_ID} \) helps avoid “routing loops” as follows: a master identifies different commands through Transaction ID (i.e., \( \text{Tran_ID} \)) and discards duplicate ones. \( MAC \) (Message Authenticated Code) is derived from the RK of ripple level \( i \). Intermediate masters that do not belong to Ripple \( i \) cannot tamper with the above
message without detection since they do not know $RK_i$. After each master receives an authenticated message, it will use the Zone Key to broadcast to all sensors in its zone.

If a message arrives at the masters in the same ripple with variable delays, there could exist the following attack: a master $A$ in ripple $i$ that receives a message $MSG$ earlier than other masters in the same ripple can fake a new message $MSG'$ and generate a new $MAC$ using its $RK_i$, and then send it to another master $B$ in ripple $i$. If $B$ has not received $MSG$ yet, it will be unable to identify the forged message $MSG'$. To protect the system from this attack, we use our aforementioned Inter-Master Pairwise key (MPK) as follows:

1. We require that each master in ripple $i$ maintain a few MPKs in its cache (called ripple-MPKs) with its close upstream and downstream masters that are located in ripple $i−1$ (upstream) and $i+1$ (downstream), respectively. We use received signal strength (RSS) to determine the geographically close masters according to the following relationship between the RSS and the distance $x$ from the transmitter [5]:

$$RSS_{db} = -10\gamma \times \log(x),$$

where $\gamma$ is the propagation path-loss coefficient [5].

2. Each time a master in ripple $i−1$ receives a message and finds out that it is supposed to be received by its next ripple, it will encrypt it by the corresponding downstream ripple-MPKs and relay the following message to the downstream masters (in ripple $i$):

To downstream $i|E(Ripple_{MPK}, Cmd_{MSG}).$

3. When a master receives the above message, it will try each of its upstream ripple-MPKs to recover the contents of the original message, i.e., $Cmd_{MSG}$.

The above scheme can prevent message spoofing from masters in the same ripple since a master only receives and decrypts messages from its upstream masters.

Since the message encrypted by Ripple Keys (RK) are transmitted in a hop-by-hop way, nodes on the smallest circle (i.e., closest to the actuator), will take large relaying load and will die more quickly than others. This is a common problem in the uplink (sensors-to-actuator/sink) transmission of any sensor network routing protocols. We will put this issue as part of our future work. However, in the downlink (actuator-to-sensors), we can simply use the broadcast scheme since an actuator has enough antenna power to cover its entire domain.

4.5. Detail low-level security procedure

In our security protocol, we use three types of messages: setup messages, data messages, and system messages for different functions. Setup messages are used for tasks that pertain to setting up the system, such as: node/CH addition, authentication, key refreshing. Data messages are used for sending and receiving data as well as generating data queries. System messages are generally used for tasks that affect the system or its topology, for example: node removal. A breakdown of all the messages within our security protocol can be seen in Fig. 11. Every message sent passes through the routing and security layer, where it gets a routing header that includes: (1) Message Source; (2) Message Destination; (3) Number of Hops; (4) List of Hop IDs.

As an example, we briefly mention two messages to be used in our secure routing discovery protocol (discussed later): (1) “RREQ” message: when a node needs to communicate with a node for which it does not have an entry in its routing table, that node sends a Route Request message (RREQ) to all of its neighbors. The neighbors then forward the RREQ to their neighbors, each appending their node ID. This process continues until the RREQ is received by the destination, which replies to the first RREQ message received. (2) “RREP” message: Route Reply messages (RREP) are generated in response to an RREQ, by the destination node, or a node that knows a route to the destination. An RREP contains the complete route from source to destination. This message then follows the path that it specifies back to the requesting node (the source). Unlike RREQs, RREP messages are eavesdropped by all nodes that receive them and the route in the RREP is extracted in order to fill in the intervening nodes routing table. This eavesdropping takes place to reduce unnecessary routing communication.

Recall that in Section 3.1, we have made five basic assumptions. Based on those initial keying information (such as initial global key, initial pairwise key and sensor ID), we can securely establish
our ripple-zone-based routing architecture (Fig. 4, Section 3) as follows:

This phase of the protocol is responsible for setting up the communication routes for inter-cluster and intra-cluster routing. A diagram depicting an overview of the route establishment phase can be seen in Fig. 12.

We have created a clustering algorithm [32] to select some sensors in an actuator domain to become cluster-heads (CHs). Once a CH is authenticated using initial pairwise key by the base station, it broadcasts an advertisement, encrypted with the initial global key. Advertisements contain the cluster ID number, and the assigned cluster key by the base station. Nodes listen to these advertisements and record their RSSs. The strongest recorded RSS is associated with the nearest CH, and the node sends a cluster join message to this CH, encrypted with the cluster key. The cluster key is received through the cluster-advertisement message (Fig. 11).

The BS keeps track of network topology through the cluster member registry of each CH. Whenever there is a change in the topology of a cluster, a new cluster organization report is sent to the BS. This knowledge is used in the event of CH compromise in order to re-organize the cluster.

After clusters are organized based on our clustering algorithm, but before the CH sends its first cluster organization report, CHs find a route to the BS. If the BS is not one of its neighbors then the CH

Fig. 11. Messages exchanged between nodes.

Fig. 12. Overview of secure route establishment phase.
broadcasts a Route Request (RREQ) message. A neighbor is defined to be a node who’s Received Signal Strength (RSS) is above a certain threshold, and every hop of every multi-hop route must occur between neighbors.

If the current recipient is not a neighbor of the requested destination, then it forwards the RREQ to all of its neighbors through a broadcast encrypted with the system key. It appends its own node ID to the route contained within the RREQ before forwarding the message. In the event that the RREQ is intended for one of the neighbors of the current recipient, the modified RREQ is forwarded only to the destination.

After the above cluster-based routing architecture is established in each actuator’s domain, every a certain time (the timer is determined by the actuator mobility and security requirements, for instance, a shorter timer can make the security performance higher), the actuator will broadcast a new global key to its entire domain encrypted by its BK (see Section 3.1). It will also broadcast new Master-to-Actuator (pairwise) keys encrypted by the corresponding last Master-to-Actuator keys to each CH in order to refresh the secure communication keys between each CH and the actuator.

4.6. Dynamic security scheme

4.6.1. Sensor addition

Sensors trying to add themselves to an existing WSAN need to undergo authentication using their initial global key, initial pairwise key and sensor ID (refer to Section 3.1 security assumptions). The BS is in fact pre-loaded such information before a new sensor joins. Once authenticated, the node broadcasts a cluster joining message, encrypted with the initial global key. CHs reply to the request with their cluster ID and cluster key, encrypted with the latest BK. The BS then helps forward the CHs response to the new sensor using the initial pairwise key with the sensor. The sensor keeps track of the RSS of all replies and joins the cluster with the highest RSS (i.e., the closest cluster). The node ID of the new sensor is then added to the cluster member registry of the CH which is eventually forwarded to the BS (in the cluster organization report phase).

4.6.2. Sensor death

When a node’s available power drops below a certain threshold, that node sends a Node Death message to its CH. The CH then sets a timer to three times the predicted remaining lifetime of the node. Once this timer expires, the CH removes this node from its cluster member registry and broadcasts a notification to its cluster. This message instructs all nodes in the cluster to eliminate the dying node from their routing tables.

4.6.3. New CH identification

A common sensor can request to become a CH if its CH is close to out of power. Once a joining CH has been authenticated by the BS through the latest Zone key and Master-to-Actuator key, it proceeds to set up a cluster. The CH broadcasts a Cluster-Advertisement message that is encrypted with the last SK (Session Key, see Section 3.1). Any node that receives a stronger signal from the new CH compared to the signal from their current CH replies with a cluster joining message, encrypted with the last SK. This message is broadcast and contains the node ID and the CH ID. This enables the node’s current CH to remove the node from its cluster member registry, and the new CH adds that node to its registry. Eventually the BS is notified of this change through a cluster organization report.

4.6.4. CH death

CH death is handled in a similar fashion to node death. When a CH reaches a certain power level, it sends a message to the BS notifying it of its state. The BS sets a timer for three times the predicted remaining lifetime of the CH. When the timer expires, the BS uses a broadcast to notify the corresponding cluster that its CH has died. This causes the nodes in the cluster to re-organize themselves and appoint the node with the largest remaining power to be the new CH.

5. Security analysis

5.1. Protection from various WSAN attacks

Now we analyze the protection of a WASN through the above security scheme from various attacks.

1. BK attacks among actuators: Because the distribution of new SKs depends on control packets encrypted by BK that is managed by group security schemes [34], it is possible for an attacker to compromise the current BK and thus can attack any future SK disclosures. Our scheme can minimize the impact of this attack through our buffered
2. **SK attacks among sensors in each actuator domain**: The attacker may modify the transmitting SK, inject phony SK, or use wireless channel interference to damage control packets. These attacks may also result in data replay and Denial-of-Service (DoS) attacks. Our scheme can easily defeat these attacks through implicit authentication. Thanks to the one-way characteristics of the hash function keys, any false SKs cannot pass the authentication test, that is, after \( L \) times \((L \leq n)\) of using hash function, if we still can NOT satisfy the following formula, we will regard that it is a false SK: (in the following formula, \( SK_{\text{FAKE}} \) is a false SK and \( SK_{\text{NOW}} \) is the SK currently used)

\[
H(H(\cdots (H(SK_{\text{FAKE}}) \cdots ))) = SK_{\text{NOW}}
\]

We have also included a rule into our security protocol: whenever a sensor continuously fails the authentication test for three times, it will send back an attack-detection message to its actuator to trigger a new High-Level initialization procedure.

3. **Relay attacks** cannot succeed because a sensor ensures that a new received SK will not only pass the authentication test but also be unique.

4. **Periodical attacks**: possibly an attacker can issue an attack at fixed time if he/she knows the value of \( \Delta' \) (i.e., the re-keying period). We address this problem through the addition of a random time interval \( \varepsilon(t) \) to \( \Delta' \), i.e., the new re-keying period is \( \Delta' + \varepsilon(t) \), where \( \varepsilon(t) \leq \Delta'/8 \).

5. **Main-in-the-middle attacks**: our scheme can also defeat Main-in-the-middle attacks (where an attacker fools the High-Level nodes as if he/she were a legal BK actuator) through a transmission of MAC in the High-Level initialization procedure as follows:

\[
\begin{align*}
\text{Sink} & \rightarrow \text{actuator} : E_{BK_{\text{new}}} \\
&A' | n | SK_0 | MAC(A' | n | SK_0))
\end{align*}
\]

6. **Data-level attacks**:

- our scheme defeats it through SK re-keying every \( \Delta' \), and inclusion of Sensor ID and per-packet IV (which will also be updated from packet to packet) in the generation of keystreams to counter the keystream-reuse problem.

6. **Performance simulations**

6.1. Jist + SWANS based WSAN security simulation

The WSAN security performance analysis results have been obtained through a Java-based simulation engine. (Section 7 will further describe our hardware test results). The simulation engine is comprised of JiST [38] (Java in Simulation Time), and SWANS [39] (Scalable Wireless Ad hoc Networks Simulator). JiST was created to simulate time in Java, while SWANS was created to simulate WSANs.

To reduce keying information loss, besides the aforementioned local loss recovery scheme (Section 4.2), we have also implemented a reliable Transport Layer protocol above our cluster-based 2-level routing protocol (see Fig. 13). We implement a CH-by-CH NACK (Negative ACKnowledge) based reliability scheme. A gap in the sequence number of sent packets indicates packet loss. Each CH maintains a list of missing packets. When a loss is detected, a tuple containing a source ID and sequence number of the lost packet is inserted into this list. Entries in the “missing packets” list are piggybacked in outgoing transmissions, and CHs infer losses by overhearing this transmission. CHs keep a small cache of recently transmitted packets, from which a child can repair losses reported by its last hop CH. Besides CH-by-CH loss recovery, we also implement end-to-end recovery. This is because heavy packet losses can lead to large missing packet lists that might exceed the memory of the motes. Our end-to-end recovery scheme leverages the fact that the base station has significantly more memory and can keep track of all missing packets. The base station attempts CH-by-CH recovery of a missing packet. When one of the CHs notice that it has seen a packet from the corresponding source, but does not have a cached copy of that packet, it adds that recovery request to its missing packets list. This request is propagated downward in this manner (using the same mechanisms described for CH-by-CH recovery) until it reaches the source. Since the
source maintains generated packets in its memory, it can repair the missing packet. 

Fig. 13 also shows that our WSAN simulator considers the actuator mobility, which causes topology change and has a direct impact on the setup of re-keying period (Section 3.1).

Fig. 14 shows our generated Ripple-zone (Section 3.1) routing topology. Our simulator can simulate a large-scale WSAN with over 500 nodes and produce simulation results within less than 1 hour, which is quicker than OPNET [41].

Energy model: SWANS [39] does not natively have a function or layer that tracks energy consumption, which is one of the most important metrics in WSANs. A battery layer was then added in order to keep track of energy consumption during simulation runtime. We use the same radio model as discussed in [4] which is the first-order radio model. In this model, a radio dissipates $E_{\text{elec}} = 50 \text{ nJ/bit}$ to run the transmitter or receiver circuitry and $I_{\text{amp}} = 100 \text{ pJ/bit/m}^2$ for the transmitter amplifier. The radios have power control and can expend the minimum required energy to reach the intended recipients. The radios can be turned off to avoid receiving unintended transmissions. An $r^2$ energy loss is used due to channel transmission [5]. The equations used to calculate transmission costs and receiving costs for a $k$-bit message and a distance $d$ are shown below:

\[
E_{\text{Tx}}(k, d) = E_{\text{Tx-elec}}(k) + E_{\text{Tx-amp}}(k, d),
\]

\[
E_{\text{Rx}}(k, d) = E_{\text{elec}} * k + E_{\text{amp}} * k * d^2.
\]

(10)
Receiving:
\[
E_{Rx}(k) = E_{Rx-elec}(k),
\]
\[
E_{Rx}(k) = E_{elec} * k. \quad (11)
\]

Receiving is also a high cost operation, therefore, the number of receives and transmissions should be minimal. In our simulations, we used a packet length \( k \) of 2000 bits. With these radio parameters, when \( d^2 \) is 500 m\(^2\), the energy spent in the amplifier part equals the energy spent in the electronics part, and therefore, the cost to transmit a packet will be twice the cost to receive.

6.2. Performance test results

Because most of WSAN nodes are tiny sensors, the security protocols should have low communication energy consumption. We have investigated the energy-efficiency of our RZ-based (Section 3) security scheme. Fig. 15 clearly shows that our scheme has the lowest energy consumption (within all nodes within a unit time) compared to the security based on other routing schemes such as Pebblenet (that is based on a simple one-level cluster scheme) [31] and the security scheme based on flat-topology routing strategy (we used the scheme in [8]).

An important reason to attribute the above energy consumption performance differences among three schemes lies in the integration schemes between the security and routing protocols. (1) As we stated in Section 1 (on our contributions) and in Section 4.5 (the integration of security with RZ-based clustering), the re-keying occurs in a hierarchical way: Inter-Zone and Intra-Zone. Most sensors (i.e., cluster members) only build secure channel with its one-hop CH, which does not require long-distance wireless transmission (and thus save energy consumption). Each CH actually only communicates with its direct upstream and downstream CHs in neighboring ripples, which limits the wireless security protocols within one hop away. All the above concepts largely shorten the wireless communication range and reduce the keying exchange times. The net effect is saving power. (2) Pebblenets [31] uses one-level clustering scheme. Each CH needs to exchange keys with all the remaining backbone CHs. It thus involves long-distance communications and large amount of keying packet transmissions, which leads to larger power consumption than our scheme; (3) the flat-based scheme does not use any clustering scheme and requires all sensors communicate with the global nodes (while we could limit the sensors communicate only inside one cluster). It is not surprising to see that flat-based security consumes most energy.

Fig. 16 shows that (before protocol optimization) a sensor that is selected as CH (Cluster Head) can lose much more energy than common sensors in its cluster. This is due to the heavy traffic load

---

**Fig. 15.** Sum of energy consumption in all nodes.

**Fig. 16.** Energy consumption in CHs and common sensors.
handled by a CH such as data aggregation results from all cluster sensors, inter-cluster communications (between two CHs), cluster key management, etc. Therefore, our RZ-based routing/security protocol adopts CH role rotation strategy, i.e., every a certain time, another cluster member will replace the role of the current CH. As shown in Fig. 17, we randomly picked up three clusters and measure the sum of remaining energy in all nodes of each cluster. All clusters have similar energy consumption amount.

In each actuator’s domain, the selection of number of clusters has a direct impact on the system energy consumption. As shown in Fig. 18, there could exist an optimal value of the No. of clusters. This can be explained as follows: when we form more clusters, more energy consumption comes from the inter-cluster communication and the high-level key management (BKs and SKs). On the other hand, when forming less clusters, the multi-hop communication overhead in each cluster (i.e., intra-cluster routing) becomes intolerable.

Based on the above-mentioned energy model (Section 5.1), we have collected the statistical results on the energy consumption of different operations in our WSAN security protocol (see Fig. 19). Most of the energy (86%) is consumed in data transmission. The local procession (such as hash function calculations) uses much less energy (only 4%) than wireless communications (86%), which implies the importance of reducing security/routing overhead in WSANs. As shown in Fig. 19, Our WSAN scheme has low-complexity due to its two-level ripple-zone topology management and symmetric cryptography protocol (the routing protocol overhead is 7% and the security key management overhead is only 3%).

We further investigate the effect of encryption key size has on power consumption. As can be seen in Fig. 20, the effect of key size on power consumption of all nodes fits exponential distribution. Increases in the level of key size can double the power consumption required for encrypting a packet. However, as shown in Fig. 19, the amount of power consumed for encryption (3%) is fairly minimal compared to the overall battery capacity. Therefore, even though higher levels of encryption require far more power, the power consumed by encryption (3%) is far less than the power consumed by communication overhead (86% + 7% = 93%).

Reliability test: we adopt both ripple-to-ripple and end-to-end loss recovery scheme to handle
packet losses (see Sections 4.2 and 5.1). Fig. 21 shows that other security schemes such as Pebblenet [31] that uses only end-to-end recovery or one-level security scheme [8] that uses a simple flooding-based recovery, have a much higher keying packet loss rate than our security scheme that is based on RZ architecture (Section 3.1).

6.3. Security overhead analysis

6.3.1. Memory storage overhead

In the proposed two-level security scheme, the high-level re-keying needs to run periodically among all actuators and the WSAN sink (i.e., the Base Station). Even though we need to store $N$-size key buffer in each actuator and also need some temporary registers for storing security parameters, memory overhead is not a big issue due to the assumption of large memory in actuators.

In low-level security, each cluster member (common sensor nodes) needs only to store in memory the sensor-to-CH key since it needs only to securely communicate with its Cluster Head. Of course each sensor should have a global keys to initialize its security with the help of the BS. Each CH needs to store M1 CH-to-CH keys with all of its neighbor CHs (M1 is determined by the clustering algorithms and sensor density), one Cluster Key, a ripple key, and M2 CH-to-sensor keys (M2 is the No. of members in its cluster). Assuming that the each of the above is 5 bytes long (40 bits), each sensor member in the cluster has to store only 6 bytes of information. The CH node needs to store $5 \times (M1 + M2 + 2)$ bytes of information. In other words, the memory overhead of the CH depends on the density of the network, the particular clustering protocol used and the frequency of re-keying protocols. If we assume a medium-size WSAN, let $M1 = 5$ and $M2 = 30$, the CH has to store 185 bytes in memory. Current Xbow Inc. Mica2 motes [40] have 4 K bytes of memory for key storage and protocol purpose. With time goes, memory could be increased at a faster speed than the battery technology. The above analysis shows that memory requirements are not an issue in our scheme. Furthermore, we can see that the vast majority of sensors (i.e., members in the clusters) need to store just a few keys. Many clustering schemes propose the concept of CH rotation, which can further decrease the possibility of memory tense in the CH nodes.

6.3.2. Security calculation overhead

We have used a first-order Markov Chain model to analyze the calculation/communication overhead when incorporating our security features into actuator communications. As calculations involving the one-way hash function in local sensor processing, consume the most energy [9]. We therefore focus on the cost of computing hash functions during each re-keying session. An actuator may fail to receive a new session key, or it may receive an incorrect session key that cannot be authenticated by using the hash function. Incorrect session keys may come from opponents attempting Denial-of-service attacks.

If the key chain buffer length is $n$, the probability of key loss is $P_{Loss}$, and the probability of key corruption is $P_{Corruption}$. We derive the expected times for hash function calculations in a re-keying cycle, $E_{Re-keying/#_of_hash}$, as follows [42]:

$$E_{Re-keying/#_of_hash} = \frac{2.5 - P_0}{2 - P_0} \left\{ \sum_{i=1}^{n} (i \cdot P_0^i) + (1 - P_{failure}) \cdot \sum_{i=0}^{n-1} (i \cdot P_0^i) \right\}, \quad (12)$$

where $P_0 = P_{Loss} + P_{Corruption}$. 

---

Fig. 21. Reliability test.
Assuming $P_{\text{Corruption}} = 0.25$, we compare the simulation and analytical results of $E_{\text{re-keying}} [# \text{of hash}]$ while varying $P_{\text{Loss}}$ from 0.0 to 0.5. Fig. 22 indicates the validity of our analytical model [42].

7. Hardware experiments

7.1. An experimental security testbed

We have used Crossbow sensor motes [40] to build our WASN hardware platform and have carried out a series of experiments to verify the efficiency of our RZ routing based security scheme. A WASN node includes two parts [40]: (1) microprocessor plus radio board (for sensor local processing, and wireless transmissions). It is also called “mote”; and (2) Sensor board (for detecting light, temperature, humidity, sound, and other types of data). Typically, these sensor motes have extremely low power (a few tens of milliwatts versus tens of watts for a typical laptop computer). When operating at 2% duty cycle (between active and sleep modes), we can achieve a lifetime of about 6 months (on a pair of AA batteries). We use MicaZ motes [40] that have IEEE 802.15.4 Physical/MAC layers modules. A Crossbow mote has a 4 MHz, 8 bit Atmel microprocessor. To build a WSAN prototype, we used a few MicaZ motes (Fig. 23 top) as the high-energy actuators, each of which controls a cluster-based routing management in its domain (Fig. 23 bottom). In each cluster, the CH and common sensors are implemented by Mica2Dot motes [40] that have less energy storage than MicaZ motes [40]. We have also implemented a peer-to-peer routing among actuators and cluster-based routing around each actuator’s domain.

Security implementation: A stream cipher RC4 has been used to implement encryption/decryption between two MicaZ nodes since the stream cipher has a lower complexity compared to a block cipher. To address the keystream-reuse problem, a sender includes its own sensor_ID into the generated keystream. For each message sent, the sender increments its own per-packet initialization-vector (IV) by 1. Keystream uniqueness can therefore be ensured. The cryptographic procedure will follow the function components as shown in Fig. 24. The Message Authentication Code (we denote it as...

![Fig. 22. Security overhead.](image)

![Fig. 23. Hardware setup: cluster-based topology setup.](image)
MAC (in Fig. 24) is included for authentication purposes. To generate multiple keys, the Pseudo Random Functions (PRF) \( f \) are adopted to derive new secret keys based on the current Session Key \( SK_{now} \) and a random number \( x \) as follows: \( KEY_{NEW} = f(SK_{now}, x) \), the generation of an \( x \) is based on the counter approach in [9].

We measure the WSAN lifetime (the time until the first sensor failure occurs due to running out of battery) in the following three scenarios: (A) using our scheme, i.e., the wireless communications occur in two levels: (1) high-level – among MicaZ motes (actuator–actuator); An actuator does not talk with a sensor belonging to another actuator’s domain; (2) low-level: including inter-cluster and one-hop intra-cluster communications. (B). flat routing topology (i.e., all sensors organize into an ad hoc network without clustering); (C) PebbleNet [31]: all sensors organize into one-level cluster architecture without the backbone consisting of actuators. We let all sensors continuously work at a high sensing frequency (50% duty cycle). Fig. 25 shows that our proposal brings over four times longer lifetime than flat-topology.

7.2. Accurate energy consumption measurement

On the controversial key management issues, there is some debate regarding the ability of sensor networks, or even actuator networks to maintain, process, distribute and update keys, and still maintain enough battery power for a significant lifetime. Our previous Jist-based WSAN simulation results (see Figs. 15, 16 and 19) have clearly shown the energy efficiency of our proposed security schemes. The hardware results (Fig. 25) also shows the lifetime extension through our security protocols.

To better observe the quantitative energy consumption benefits from different security schemes, we need to adopt a good sensor network energy measurement model that is suitable to different application scenarios. Coarse approximations of the energy consumption are usually derived from the number of transmitted packets and CPU duty cycles (as we did in Section 6). Such approximations may fail to capture low-level details and states of a device. For a quantitative evaluation, a precise and
detailed low-level model of the device is needed. Although recent research provides many energy-efficient or energy-aware applications, only a very limited number (such as [43]) has been evaluated deeply in terms of quantitative energy consumption by measurements of current draw. For hardware development, some energy profiling tools have been presented (such as [44]), focusing on modeling the energy consumption of processors and microcontrollers for embedded systems as they add models for the memory, serial communication, and other parts of the microcontroller. These tools mainly focus on hardware development and not on the evaluation of software for distributed systems. None of the tools include models for devices next to the microcontroller like communication, external memory, sensors, and actuators. Thus, they can only be used for the evaluation of single separate components without inter-device communication and interaction with the environment interaction. However, as communication and sensing are the main purpose of sensor nodes, these tools are only of limited use for energy evaluation in sensor networks [45].

Recently, Power Tossim [46] has been presented as an extension to the TinyOS [48] simulator Tossim [47] to estimate the energy consumption of the Mica2 sensor node. In PowerTOSSIM, TinyOS components corresponding to specific hardware peripherals (such as the radio, EEPROM, LEDs, and so forth) are instrumented to obtain a trace of each device’s activity during the simulation run. Through the outputted realtime traces of current draw in our RZ-based security and Pebblenets scheme [31], we have collected the energy consumptions of major components (such as CPU idle, CPU active, Radio, etc.) in Mica2 motes as in Table 1. While active CPU energy (when executing actual instructions) is very small, the total CPU energy consumes a significant fraction of the energy for the entire run.

### Table 1
PowerTOSSIM Energy Consumption comparisons

<table>
<thead>
<tr>
<th>(in mJ)</th>
<th>RZ-based security</th>
<th>Pebblenets [31]</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Idle</td>
<td>651.87</td>
<td>715.53</td>
</tr>
<tr>
<td>CPU active</td>
<td>25.32</td>
<td>45.21</td>
</tr>
<tr>
<td>Radio</td>
<td>1002.11</td>
<td>1542.76</td>
</tr>
<tr>
<td>LEDs</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Sensor board</td>
<td>65.86</td>
<td>63.21</td>
</tr>
<tr>
<td>EEPROM</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Total</td>
<td>1745.16 mJ</td>
<td>2366.71 mJ</td>
</tr>
</tbody>
</table>

This is because two security applications leave the CPU in the 3.2 mA idle state rather than using the lower-power sleep modes.

From Table 1, we can easily see that for the two most important components, i.e., CPU active and Radio transmission, our proposed scheme shows significant improvements over general cluster-based Pebblenets scheme (the energy saving efficiency is improved by 78% and 54% respectively).

We have also performed multiple rounds of measurements on the energy consumption before and after we add the proposed security schemes. As seen in Table 2, the average energy overhead increase after adding security to the routing schemes is less than 5%, which clearly indicates the feasibility of using symmetric-key based cryptography in WSANs.

### 8. Conclusions

The focus of this paper is the security design in an important information infrastructure – large-scale and low-energy Wireless Sensor and Actuator Networks (WSANs). Our schemes attempt to ensure that data is transmitted among actuators and sensors with desired security (i.e., overcoming external network attacks). We have proposed to seamlessly integrate WASN security with a promising routing architecture that is scalable and energy-efficient. To protect from active attacks in sensor networks, we have developed two-level re-keying schemes that cannot only adapt to a dynamic network topology but also securely update keys for each data transmission session. Moreover, to provide the security for the in-networking processing such as data aggregation in WASNs, we have defined a multiple-key management scheme in conjunction with the proposed Ripple-Zone (RZ) routing architecture. In the future, other trustworthiness issues in WSANs such as broadcast authentication with the assumption of time synchronization and countermeasures for Sybil attacks [2] in mobile actuators need further investigation.
References


[20] Yee Wei Law, Sandro Etalle, Pieter H. Hartel, Key Management with group-wise pre-deployed keying and secret sharing pre-deployed keying, EYES project (Europe), available through GOOGLE search engine.


Fei Hu is currently an Assistant Professor in Computer Engineering Department at RIT, NY. He received his Ph.D. degree in the Department of Electrical and Computer Engineering at Clarkson University in 2002. His Ph.D. research was on high-performance transmission issues in wireless networks. He obtained his B.S. and M.S. degrees from Shanghai Tiedao University (China) in 1993 and 1996, respectively. He served as a Senior Networking Engineer in Shanghai Networking Lab and Shanghai Lucent Inc. from 1996 and 1999 where he worked on several large projects on high-performance networks. He is a Full SigmaX Member and IEEE chapter officer. His research interests are in ad hoc sensor networks, 3G wireless and mobile networks and network security.

Waqaas Siddiqui is a graduate student in Computer Engineering department at RIT under the supervision of Dr. Fei Hu. His research focus is security in wireless sensor networks through a scalable routing architecture.

Krishna Sankar is currently with Cisco Systems as a Member Of Technical Staff in their Global Commerce Technology division. He has about 20 years of experiences ranging from Software Architecture and Development to Industrial Engineering to Author, Speaker, Entrepreneur and Technology Evangelist. He has worked with many organizations including US Air Force, Navy, HP, Qantas, Air Canada and Ford. His technology interests include XML and web services standards, Peer-To-Peer and distributed security, Linux kernel security, web service/web process networks and e-commerce – dynamic configurable multi-partner trading networks.