Secure Key Distribution Using Fragmentation and Assimilation in Wireless Sensor and Actor Networks

AtaUllah Ghafoor, 1,2 Muhammad Sher, 2 Muhammad Imran, 3 and Abdelouahid Derhab 3

1 Department of Computer Science and Software Engineering, International Islamic University, Islamabad 44000, Pakistan
2 Department of Computer Science, National University of Modern Languages, Islamabad 44000, Pakistan
3 King Saud University, P.O. Box 92444, Riyadh 11543, Saudi Arabia

Correspondence should be addressed to AtaUllah Ghafoor; atullah4us@gmail.com

Received 12 December 2014; Accepted 16 February 2015

Copyright © 2015 AtaUllah Ghafoor et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Secure key distribution is one of the most important building blocks to ensure confidentiality in wireless sensor and actor networks (WSANs). However, the key distribution becomes a challenging task when keys have to be relayed by compromised nodes in order to establish secure communication between distant nodes. To deal with this issue, we propose in this paper a key distribution scheme, named key distribution using fragmentation and assimilation (KDFA). In this scheme, the sender node splits the actual key into fragments and sends them through intermediate actor nodes towards the receiver node. The latter assimilates these key fragments using XOR operation to reconstruct the actual key. In this case, attacker cannot retrieve the actual key and only gets the key fragment. KDFA is composed of two parts: (a) key distribution protocol (KDP) to distribute the key using intra-actor and interactor communication scenarios and (b) key fragmentation algorithm (KFA) to slice the key using binary calculations. KDFA execution has been specified and formally verified using Rubin logic. Moreover, the performance and the resilience of the proposed protocol are further analyzed through extensive simulations using ns-2.35. Results show that KDFA is resilient against actor node compromise and key exposure.

1. Introduction

A wireless sensor and actor network (WSAN) is a distributed system of sensor and actor nodes that are interconnected through wireless links [1, 2]. The sensor nodes gather information about the environment such as temperature and vibration and send them to a base station (or sink) using a hop-by-hop communication. On the other hand, the actor nodes perform actions in response to some specific events. Such networks are widely used in applications like buildings/house monitoring [3], water sprinkling on fire detection [4], surveillance and intruder detection, traffic monitoring, health and habitat monitoring, and so forth [5, 6]. Actor nodes play a vital role at taking actions to change the behavior of the environment by analyzing and responding to the events [7]. They also communicate with ordinary sensor nodes to collect data as shown in Figure 1.

Secure key distribution in WSAN is mandatory because most of the applications require confidentiality of data exchanged between nodes located in different regions [8, 9]. In the literature, some secure key establishment schemes have been proposed to enhance security and ensure confidentiality. In existing symmetric key management schemes [10–12], if a node is compromised then the new key distribution messages passing through that node are also compromised. Therefore, the links are compromised before their establishment, and it is not possible to establish end-to-end key between the sender and the receiver as messages can be decrypted and then reencrypted at the intermediate nodes. An intermediate malicious node can interpret these keys and help other malicious nodes at joining the network using valid keys. The problem becomes more complex when indirect keys are exchanged between distant nodes in different regions because more intermediate nodes are involved and the probability of the key to be relayed by malicious intermediate nodes increases.

In the existing key distribution schemes, all the keys that pass through compromised nodes also become compromised. The issue studied in this paper is how to exchange
Figure 1: Communication scenarios in WSAN.

secret keys among distant nodes in the presence of malicious intermediate nodes. To deal with this issue, we propose a key distribution using fragmentation and assimilation (KDFA) scheme for establishing indirect keys between two distant sensor nodes. The main contribution offered by KDFA is that the sender node slices the actual key into fragments and sends them via intermediate actor nodes towards the receiver node. The distant receiver node assimilates the key fragments using XOR operation to obtain the original key. In this case, the attacker can only get the key fragment because the original key is never transmitted. The condition for successful attack is to capture all the fragment key packets related to one key, which is not a trivial task. If an attacker succeeds at identifying the intermediate nodes and subverting the key fragments then only one key is exposed. Therefore, KDFA is expected to provide better secure key distribution scheme and achieve better resilience against malicious nodes in comparison to other schemes.

KDFA is composed of two parts: (a) key fragmentation algorithm (KFA), which divides the keys of different sizes into an appropriate number of key fragments, and (b) key distribution protocol (KDP) provides a stepwise description for exchanging secure messages to distribute pairwise keys. In addition, we have formally specified and verified KDFA using Rubin logic [13]. The latter validates the protocol as per standard requirements of cryptographic operations like authentication, message integrity, message freshness, encryption, decryption, and so forth. It also helps at identifying the deficiencies in the protocol and possibilities of node compromise attacks, and it is also near to real implementation. KDFA is further simulated under ns-2.35 to assess its effectiveness in terms of resilience against compromised nodes. Results show that two uncompromised nodes can distribute a key securely even if a large number of intermediate nodes are compromised.

The rest of the paper is organized as follows: system model and problem statement are provided in Section 2. Section 3 presents related work. Section 4 details KDFA operations for key distribution. Section 5 provides a detailed formal specification and analysis of KDFA using Rubin logic. Simulation scenarios and results in terms of resilience, fragmentation computation, and communication overhead are given in Section 6. Finally, Section 7 concludes the work and outlines future research directions.

2. System Model and Problem Statement

We consider a model of WSAN, in which the nodes composing the networks (i.e., sensors, actors, and the sink) are randomly deployed in the network field. We assume that each actor knows the keys of all the sensor nodes in the network and can move around within the network field to collect data from sensors. We also assume that nodes within communication range of each other have already established keys among them.

During indirect key establishment between ordinary distant nodes, the sender node transmits a pairwise key via intermediate nodes. The message that contains a new key is encrypted by the preestablished key between the sender and the intermediate node. The latter decrypts the key and then reencrypts it using the preestablished key with the next intermediate node and so on. By following the same process, the new key reaches the receiver node. The problem in this scenario is that any new secret key is available to intermediate nodes in a plain text form. If one of the intermediate nodes is compromised then that key is compromised before its establishment. Similarly, future key passed through that nodes become also compromised. The same problem holds in [10–12, 14–17] for WSN and [9, 16, 18–20] for WSAN.

This allows the attacker to perform illegal activities by either code alteration or falsifying the broadcast messages. New indirect keys can also be established with that malicious node. An adversary can also introduce new node to the network by preloading the compromised keys onto it along with a duplicate ID of a sensor node that was expired by the attacker. This new duplicated node can receive the packets destined for the actual node. In a similar way the situation can be worse if a large number of new compromised actor nodes are introduced to the network.

3. Related Work

Key management schemes have been extensively investigated in the context of indirect key establishment between nodes in WSNs and WSANs. In this scenario, a symmetric key is transmitted by the sender node to the receiver node using intermediaries. Intermediate nodes in the literature are called agent [15], gateway [10], bridge [12], or proxy [17]. These nodes play a vital role for intercluster communication in WSN. We will use the name gateway to represent such nodes. Also, the similar role is played by actors for indirect key establishment in WSAN [9, 16, 18–20].

3.1. Gateway Node Based Schemes. Zhou et al. [15] provided a pairwise intergroup key distribution scenario where an agent node pair is used to share keys between senders and receivers.
This scenario is depicted in Figure 2 where the sender node n2 sends the message via an agent node pair n7 and n23 to the receiver node n30. This scheme uses an agent pair to transmit a single key whereas in our proposed scheme (i.e., KDFA) actor pairs are used instead.

Addya and Turuk [10] proposed a scheme, which required the gateway node to set up an intergroup path key between nodes of two clusters. A single node is acting as a gateway node instead of a node pair because it has shared keys of both clusters. This node should be located at a common region of two clusters similar to the locations of nodes 5, 11 as illustrated in Figure 3. Intercluster communication between cluster 1 and cluster 2 is established through node 5. There could be multiple gateway nodes in a common region. The scheme uses a gateway node to transmit a single key which is similar to the intra-actor communication scheme proposed in KDFA.

Zhou and Fang [21] proposed a symmetric polynomial based scheme where polynomial shared keys are preloaded in each sensor to calculate the direct key after deployment. Nodes located at the junction of two clusters receive two polynomials and can play the role of gateway. To achieve satisfactory level of security, coefficient degree must satisfy (1)

\[ 0 \leq N_i - 2 \leq t \]
\[ \sum_{i=1}^{K} \frac{k(K+1)!}{2} \]
\[ i = 1, 2, \ldots, K. \]

In this scheme, if a new node joins the network or existing node leaves the network then new polynomial is calculated to refresh the existing keys. It requires a large amount of communication overhead. It could be reduced by adopting XOR based key freshness scheme [22] where a magic word is shared among all member nodes to refresh the key in an efficient manner.

Huáng’s scheme [14] mitigated the effects of compromised nodes by applying the Reed-Solomon encoding scheme to divide the seed of a key into \( n \) codewords and transmit each codeword via different node-disjoint path. KDFA applies the idea of partitioning on indirect keys as in [14]. However, instead of using some encoding scheme, it just uses a lightweight fragmentation mechanism to partition the key into small-sized fragments.

3.2. Actor Node Based Schemes. Du et al. [9] proposed a distributed key establishment scheme for WSAN, in which actors are preloaded with symmetric keys and public key certificates as well. After deployment, node A can request from its neighbors to sign its certificate where an actor can also be one of the signers. Neighboring nodes sign the received certificate by using their self-generated private key \( K^{-1}_N \) and return the signatures \( \{C_{ert_{A}}\}K^{-1}_N \) back to node A. Afterwards, communicating nodes can exchange a key by sharing symmetric session key encrypted using public keys. If a node is captured then its certificates are also removed from the list of neighboring nodes. It claims to achieve guaranteed security if a small number of nodes are malicious. In this scheme, nodes are relying on ordinary intermediate nodes for signing the certificates where a malicious node can also take part in establishing keys. KDFA is a reliable scheme that ensures the resilience and the secure key distribution in case a large number of nodes are compromised.

Mármol et al. [19] proposed WSANRep to maintain node reputation to detect the malicious nodes and exclude from the network. Global reputation of a network \( WSAN_N \) can be calculated by a group \( G_i \) at the \( t \)th transaction using (2) where \( \alpha, \beta, \) and \( \gamma \) are weights assigned based on previous experiences:

\[ GRep_{(G_i,WSAN_j)}^{(t)} = \left( \alpha_{(G_i,WSAN_j)} \cdot GRep_{(G_i,WSAN_j)}^{(t-1)} \right) + \left( \beta_{(G_i,WSAN_j)} \cdot Rep_{(G_i,WSAN_j)} \right) + \left( \gamma_{(G_i,WSAN_j)} \cdot Rep_{(G_i,WSAN_j)} \right). \]
Kashi and Sharifi [23] illustrated a categorical comparison for weak coordination and connectivity between actors and sensor nodes. It recommends that powerful actor nodes can help sensor nodes in routing and forwarding data to the sink. KDF provides a strong resilience to protect the key exposure at intermediate compromised nodes. Interactor connectivity is the mandatory part for this communication; therefore, connectivity restoration [24–26] should also be considered to achieve sustainable communication and coordination [27] in different regions using actor nodes.

Y. Lee and S. Lee [20] proposed a scheme to establish pairwise keys between two sensors and between sensors and actors for secure communication in WSAN. A region key is also distributed by actor nodes to its neighboring sensor nodes for a specific task and keys expire after a fixed time quanta. Public key cryptography was proposed only for upper layer communication between actors and the sink nodes. Key distribution through public key cryptography is an expensive task in terms of computation and a central KDC could be a single point of failure. KDF does not require a KDC and it ensures secure key distribution as compared to the existing schemes.

4. Key Distribution Using Fragmentation and Assimilation (KDF)

The proposed scheme KDF provides a reliable solution to the key exposure problem at compromised intermediate nodes. A sender node transmits a request to nearby actors for establishing key with a distant receiver node that is located outside the communication range. Actors can play a vital role at routing keys between sender and receiver nodes. Key distribution process begins when a sender uses a lightweight fragmentation mechanism to divide the key into $f$ fragments and transmits them through $f$ intermediate actor nodes in the paths towards the receiver. The receiver calculates the actual key by taking XOR of these key fragments. In this process, the number of key fragments “$f$” is decided by the administrator and it depends on the safety of deployment region and the security level required by the application. In most of the cases, the recommended value for $f$ is either 2 or 3. In KDF, the attacker can only get the key fragment by capturing intermediate nodes and not the complete original key. It provides an efficient way to secure the pairwise key establishment scheme by improving the resilience against the node capture attacks.

Actor nodes can distribute the key fragments using one of the following communication scenarios. In the first scenario, the sender requests to establish an indirect key with a node located outside its transmission range but exist within the communication range of nearby actor nodes. The actor node has keys of all the nodes in the network and can act as an intermediate node to establish a secret key between sender $S_x$ and receiver $R_y$. This scenario is shown in Figure 4 where $f = 2$ fragments are transmitted using actor nodes $A_x$ and $A_y$. Afterwards, $S_x$ and $R_y$ can exchange encrypted messages via ordinary intermediate sensor nodes.

In the second scenario, the sender node transmits two key fragments to the actor node pair to establish indirect key with a distant node $R_k$ that is not within transmission range of actor $A_x$ and $A_y$. Actors request the adjacent actor to distribute the key fragments to $R_k$. It requires interactor communication between two actor pairs for $f = 2$ fragments, as illustrated in Figure 5.

In the third scenario, the sender node requests from nearby actor node pair to establish indirect key with a distant node by distributing key fragments. Actor $A_x$ is within the transmission range of $R_k$ and directly sends the key fragment to $R_k$. But the receiver node $R_k$ is out of the transmission range of the second actor $A_y$, which requires interactor communication to distribute the key fragment to $R_k$. Figure 6 depicts the key establishment for the $f = 2$ key fragments using hybrid communication scenario.

During the key distribution process, if there is only one actor in the range of the sender node and the second actor node is not available within a threshold value of time then $f$ key fragment messages are sent via a single actor node. An upper-bound on the delay between consecutive transmissions of fragments can be imposed so that the
receiver node will wait for the next fragment. An extra functionality can be added to the receiver code to manage the freshness as different messages originated by the same node reach the receivers at different time instants. During the waiting phase, another actor can visit the sender node’s area to transmit other fragments because the actors frequently visit areas to collect data from the sensor nodes. In all cases, the property ensured by KDFA is that a single key is never sent in one message.

KDFA is applicable for static and mobile sensor nodes. In war field, two or more distant soldiers can exchange their current positions securely to move together towards the target. In industry, big machines are turned on in a sequential manner to avoid power hazards. Distant machines can be synchronized to turn on or off by exchanging secure messages. There exist a number of similar application scenarios where our scheme can effectively provide secure messaging.

A list of notations used in KDFA is provided with brief description in Table 1.

4.1. Key Fragmentation Algorithm (KFA). It is used by the sender node to divide the key into \( f \) fragments. If the key size is divisible by \( f \) then equal-sized fragments are produced like 32-bit key is divided into 2 fragments of size 16-bit each, but in case of 3 fragments the result will be two fragments of size 10 and the last fragment of size 12. The size of the last fragment is calculated as per line 7 in Pseudocode 1.

**Step 1, 2.** If a selected key size in bits is completely divisible by \( f \), for example, a 32-bit key with \( f = 2 \), then the remainder \( \text{Rem}(32, 2) \) is zero. It produces 2 fragments of 16-bit each.

**Step 3.** Last fragment size is equal to the fragment size because the key size is completely divisible.

**Step 4, 5.** If the key size is not completely divisible by \( f \), for example, a 32-bit key with \( f = 3 \). In this case, the remainder \( \text{Rem}(32, 3) \) is not zero. It will create three fragments of 10-bit each with a total of 30 bits and an extra fourth fragment of size 2-bit is also created. The last fragment is smaller than the selected fragment size, which is 10 in this case.

**Step 6.** This problem is mitigated by calculating additional bits and then padded them to the last fragment. Initially a fragment size is calculated by dividing the key size by \( f \) and the floor is taken to get the lower value for the fragment size. If a key size is 32 with \( f = 3 \) then the fragment size is 10. It will create \( (f - 1) \) fragments of (fragment size \( \times (f - 1) \)) bits. It will produce two fragments of 20 bits when key size = 32, \( f = 3 \) and fragment Size = 10. Additional number of bits required for padding for the last fragment is calculated by multiplying the fragment size by \( f \) to get the total size in bits of all fragments. Then, we ignore extra bits left due to floor operation. It will result into 32 – 30 = 2 bits.

**Step 7.** The last fragment size \( LF_2 \) is calculated by adding the number of padded additional bits to the fragment size. In the case of above scenario, fragment size is 10 and additional bits are 2; therefore, the last fragment size equals 12 bits.

Bitwise representation of this technique is elaborated in the following example where a 16-bit key 110011101010101 is divided into \( f = 2 \) fragments that are 1100110000000000 and 00000000001010101. In this case the remainder \( \text{Rem}(16, 2) = 0 \) that satisfies the if condition where fragment size is equal to

![Figure 6: Hybrid communication for key distribution.](image-url)
8 bit. The original key will be obtained by taking the XOR as follows:

\[ \begin{align*}
00000000 & 01010101 \\
11100111 & 10000000 \\
00000000 & 01010101 \\
11100111 & 10000000 \\
\end{align*} \]  
\[ (3) \]

In another scenario, 16-bit key \(1100110101010101\) is divided into \( f = 3 \) fragments including \(1110000000000000\), \(0000011101000000\), and \(0000000000010101\). It is obvious that the first two fragments contain 5 key bits and 11 padded zeros whereas the last fragment contains one extra bit from the original key, that is, 6 bits of key and 10 zeros. In this case the remainder \( \text{Rem}(16, 3) = 1 \) which is not equal to zero. Therefore the else part of algorithm (i.e., line (4)) is executed where the fragment size equals 5 and the last fragment size equals 6. By applying the XOR operation on these fragments, the original key is obtained as follows:

\[ \begin{align*}
1110000000000000 \\
0000011101000000 \\
0000000000010101 \\
1110011101010101 \\
\end{align*} \]  
\[ (4) \]

4.2. Key Distribution Protocol (KDP). The sender and receiver nodes establish the secret keys using KDP protocol. In this section, all the protocol steps are elaborated along with description of message contents for actor distribution scenario. It includes sender \( S_i \), actor \( A_x \), \( A_y \), and receiver node \( R_k \). KDP starts when the sender node \( S_i \) randomly selects an actor node for sending one key fragment \( K_{F_m} \) and sends it to actor \( A_x \) encrypted with the association key it shares with \( A_x \). Key fragment \( K_{F_m} \) represents the \( m \)th key fragment where \( 1 \leq m \leq f \). \( A_x \) is the actor node where \( x \) represents the randomly selected actor ID. The value of \( x \) could be any number from \( 1 \) to \( Q \), which represents the total number of actors in the network. Node \( S_i \) will encrypt the message \((K_{F_m}, R_k, n_i, t_i, MAC(K_{F_m} \| n_i \| t_i))\) using a shared preestablished key with the actor node \( A_x \) as described in (5). In the message, \( n_i \) is nonce value, \( t_i \) is timestamp, \( MAC(K_{F_m} \| n_i \| t_i) \) hash of concatenated strings of \( K_{F_m} \) and \( n_i \) and \( t_i \) are used to ensure integrity protection of key fragment and also guard against replay attacks:

\[ C_1 = E_{K_{S_i-A_x}}(K_{F_m}, R_k, n_i, t_i, MAC(K_{F_m} \| n_i \| t_i)) \]  
\[ (5) \]

Sender node \( S_i \) sends the message (ID\(_{S_i}, C_1\)) to actor node \( A_x \). After receiving the message, actor \( A_x \) decrypts the message, as per

\[ (K_{F_m}, R_k, n_i, t_i, MAC(K_{F_m} \| n_i \| t_i)) = D_{K_{S_i-A_x}}(\text{ID}_{S_i}, C_1). \]  
\[ (6) \]

Actor \( A_x \) reencrypts the retrieved message using the association key \( K_{A_x-R_k} \) shared with the receiver node \( R_k \), as shown in

\[ C_2 = E_{K_{A_x-R_k}}(K_{F_m}, n_i, t_i, MAC(K_{F_m} \| n_i \| t_i)). \]  
\[ (7) \]

Actor \( A_x \) sends the message (ID\(_{A_x}, C_2\)) to the receiver node \( R_k \). Upon receiving the message, the receiver node \( R_k \) decrypts the message using its preestablished key with \( A_x \), as illustrated in

\[ (K_{F_m}, n_i, t_i, MAC(K_{F_m} \| n_i \| t_i)) = D_{K_{A_x-R_k}}(\text{ID}_{A_x}, C_2). \]  
\[ (8) \]

Actor \( A_y \) also reencrypts the message according to (9) by using the preestablished key between actor \( A_y \) and the receiver node \( R_k \):

\[ C_3 = E_{K_{A_y-R_k}}(K_{F_m+1}, n_{i+1}, t_{i+1}, MAC(K_{F_m+1} \| n_{i+1} \| t_{i+1})). \]  
\[ (9) \]

Actor \( A_y \) sends the message (ID\(_{A_y}, C_3\)) to the receiver Node \( R_k \). \( R_k \) then recovers \( K_{F_m+1} \) as identified in (10) by using its preloaded association key \( K_{A_y-R_k} \) established with \( A_y \):

\[ (K_{F_m+1}, n_{i+1}, t_{i+1}, MAC(K_{F_m+1} \| n_{i+1} \| t_{i+1})), \]  
\[ (10) \]

\[ = D_{K_{A_y-R_k}}(\text{ID}_{A_y}, C_3). \]

Finally \( R_k \) receives \( f \) key fragments and assimilates the pairwise key using (II) where \( \times \) represents XOR operation:

\[ K_{S_i-R_k} = K_{F_m} \times K_{F_m+1}. \]  
\[ (11) \]

For Interactor distribution scenario, \( A_x \) transmits the message to \( B_x \) and then \( B_x \) sends it to the receiver node \( R_k \). Similarly \( A_y \) transmits via \( B_y \) to the receiver node \( R_k \). Following additional messages are sent for interactor scenario:

\[ A_x \rightarrow B_x: E_{K_{A_x-B_x}}(K_{F_m}, n_i, t_i, MAC(K_{F_m} \| n_i \| t_i)) \]

\[ A_y \rightarrow B_y: E_{K_{A_y-B_y}}(K_{F_m+1}, n_{i+1}, t_{i+1}, MAC(K_{F_m+1} \| n_{i+1} \| t_{i+1}))) \]

\[ B_x \rightarrow R_k: E_{K_{B_x-R_k}}(K_{F_m}, n_i, t_i, MAC(K_{F_m} \| n_i \| t_i)) \]

\[ B_y \rightarrow R_k: E_{K_{B_y-R_k}}(K_{F_m+1}, n_{i+1}, t_{i+1}, MAC(K_{F_m+1} \| n_{i+1} \| t_{i+1}))). \]

5. KDF Formal Specification

We have performed formal modeling using nonmonotonic cryptographic protocol (NCP) also called Rubin logic [13] to analyze and verify the KDF protocol as per formal specifications. It validates the protocol as per standard requirements of cryptographic operations like authentication, message integrity, message freshness, encryption, decryption, and so forth. It also helps to identify the deficiencies in the proposed protocol and possibilities of compromise attacks. It is close to the actual implementation and flow of programming functions. In Rubin logic, the entities are assigned roles and a global set is also maintained where information about the protocol is maintained in these sets and it also maintains
Table 2: Notations of local set for KDFA.

| Sr. | Notation | Description |
|-----|----------|-------------|
| 1   | $K_{f_m}, K_{f_{m+1}}$ | First and second key fragment for $f = 2$ |
| 2   | $P_{S_{f_m}}, P_{S_{f_{m+1}}}$ | Concatenated string at $S_i$ for the first and the second key fragment |
| 3   | $M_1 - M_2$ | Messages 1 to 2 |
| 4   | $C_1 - C_3$ | Cipher texts of $M_1$ to $M_2$ |
| 5   | $H_{S_{f_m}}, H_{S_{f_{m+1}}}$ | Hash calculated at $S_i$ for $K_{f_m}$ and $K_{f_{m+1}}$ |
| 6   | $H^*, H^\ast$ | Hash of message $(K_{f_m}, n_i, t_i)$ received at $A_y$ and $R_k$ for the first key fragment |
| 7   | $H^\ast$ | Hash of message $(K_{f_{m+1}}, n_{i+1}, t_{i+1})$ received at $A_y$ |
| 8   | $H_{R_k}$ | Hash calculated at $R_k$ for reply to $S_i$ |
| 9   | $P_{R_k}$ | Concatenated string at $R_k$ for reply |
| 10  | $m_i$ | Message count at $R_k$ |

updated states of the users after each updateable operation. Global sets are accessible to all the member nodes and can be categorized into secret, observer, rule, and principal sets. A detailed discussion on formal specification for WSAN protocols is provided in [25, 28] along with appropriate case studies.

We have reutilized the notations of KDFA predefined earlier in Table 2 and a brief description is provided in Table 2 for all notations used in the following local set for formal specifications.

A local set is maintained at each entity or node and can be categorized into possession POSS(), belief BEL(), and seen and behavior sets BL(). Detailed specification of this sets can be explored from [13, 16]. By applying Rubin logic on KDFK-AN, the local set is elaborated as follows and then its verification and analysis part provides a detailed overview of all sets maintained under the category of local set.

Local Set for KDFA

(1) Sender ($S_i$)

POSS($S_i$) = \{ID$_S$, $K_{S_i-A_y}$, $K_{S_i-A_i}$\}

BEL($S_i$) = \{|ID$_S$|, $(K_{S_i-A_y})$, $(K_{S_i-A_i})$\}

BL($S_i$) =

\[\text{Hash}(h(\cdot); K_{f_m}, n_i, t_i) \rightarrow H_{S_{f_m}}\]

\[\text{Concat}(K_{f_m}, n_i, R_k, t_i, H_{S_{f_m}}) \rightarrow P_{S_{f_m}}\]

\[\text{Encrypt}([P_{S_{f_m}}, K_{S_i-A_y}] \rightarrow C_1\]

\[\text{Send}(A_y, [ID_S, C_1]) \rightarrow M_1\]

\[\text{Update}(n_i, t_i, K_{f_m})\]

\[\text{Hash}(h(\cdot); K_{f_{m+1}}, n_{i+1}, t_{i+1}) \rightarrow H_{S_{f_{m+1}}}\]

\[\text{Concat}(K_{f_{m+1}}, R_k, n_{i+1}, t_{i+1}, H_{S_{f_{m+1}}}) \rightarrow P_{S_{f_{m+1}}}\]

Encrypt([P$_{S_{f_{m+1}}}$, K$_{S_i-A_y}$] → C$_2$

Send($A_y, [ID_S, C_2]$) → M$_2$

Update($n_{i+1}, t_{i+1}, K_{f_{m+1}}$)

Receive($R_k, [ID_{R_k}, C_3]$)

Split([ID$_{R_k}$, C$_3$])

Decrypt([C$_3$, K$_{A_y-R_k}$]) to get

\[n_i^*, t_k, H_{R_k}\]

Check-freshness($t_{A_y} - t_k$) ≥ Δt

if yes, then abort

Check($n_i, n_i^*$)

MAC([Concat($n_i, t_k, H_{R_k}$)]) → H$^\ast$

Check(H$^\ast$)

(2a) Actor ($A_y$)

POSS($A_y$) = \{ID$_{A_y}$, $K_{A_y-S_i}$, $K_{A_y-A_y}$\}

BEL($A_y$) = \{|ID$_{A_y}$|, $(K_{A_y-S_i})$, $(K_{A_y-A_y})$\}

BL($A_y$) =

Receive($S_i, [ID_S, C_1]$)

Split([ID$_S$, C$_1$])

Decrypt([C$_1$, K$_{A_y-S_i}$]) to get

\[K_{F_m}, R_k, n_i, t_i, H_{S_{f_m}}\]

Check-freshness($t_{A_y} - t_k$) ≥ Δt

if yes, then abort

MAC([Concat($K_{f_m}, n_i, t_i$)]) → H$^\ast$

Check(H$_{S_{f_m}}, H^\ast$)

ConCat($K_{f_m}, n_i, t_i, H_{S_{f_m}}$) → P$_{A_y}$

Encrypt([P$_{A_y}$, K$_{A_y-R_k}$]) → C$_3$

Send($R_k, [ID_{A_y}, C_3]$) → M$_3$

Update(M$_{ID}$)

(2b) Actor ($A_y$)

POSS($A_y$) = \{ID$_{A_y}$, $K_{A_y-S_i}$, $K_{A_y-A_y}$, $K_{A_y-A_y}$\}

BEL($A_y$) = \{|ID$_{A_y}$|, $(K_{A_y-S_i})$, $(K_{A_y-A_y})$, $(K_{A_y-A_y})$\}

BL($A_y$) =

Receive($S_i, [ID_S, C_2]$)

Split([ID$_S$, C$_2$])

Decrypt([C$_2$, K$_{A_y-S_i}$]) to get

\[K_{F_{m+1}}, R_k, n_{i+1}, t_{i+1}, H_{S_{f_{m+1}}}\]
Check-freshness($t_{A_y} - t_{x+1}) \geq \Delta t$

if yes, then abort

MAC({Concat($K_{F_m, n_i, t_{x+1}}$)}) $\rightarrow H^*$

Check($H_{S,F_m}$, $H^*$)

Concat($K_{F_m, n_i, t_{x+1}}, H_{S,F_m}$) $\rightarrow P_{A_y}$

Encrypt({$P_{A_y}, K_{A_y, n_i}$}) $\rightarrow C_4$

Send($R_k$, {ID$_A$, C3}) $\rightarrow M_4$

Update($M_{ID}$)

(3) Receiver ($R_k$)

POSS($R_k$) = {ID$_R$, K$_{A_y, S_i}$, K$_{A_y, n_i}$, m$_c$}

BEL($R_k$) = {#(ID$_R$), (K$_{A_y, S_i}$), (K$_{A_y, n_i}$), m$_c$}

BL($R_k$) =

Receive($A_y$, {ID$_A$, C3})

Split({ID$_A$, C3})

Decrypt($C_3$) to get

[K$_{F_m, n_i, t_{x+1}}, H_{S,F_m}$]

Check-freshness($t_k - t_i) \geq \Delta t$

if yes, then abort

MAC({Concat($K_{F_m, n_i, t_{x+1}}$)}) $\rightarrow H^*$

Check($H_{S,F_m}$, $H^*$)

INCR($m_{c, 1}$)

Compare($m_c$ equals $f$) if NO continue

XOR($K_{F_m, K_{F_m}}$) $\rightarrow K_{R_k, S_i}$

Hash($h(); n_i, t_k) \rightarrow H_{R_k}$

Concat($n_i, t_k, H_{R_k}) \rightarrow P_{R_k}$

Encrypt($P_{R_k}, K_{R_k, S_i}) \rightarrow C_5$

Send($S_i$, {ID$_R$, C3}) $\rightarrow M_5$

Update($M_{ID}$)

Local sets for all the entities including sender, receiver, and actor nodes are separately maintained. A possession set POSS(entity) contains all the parameters involved in encryption, decryption, and other operations performed at local memory of each entity as described in section below. A Behavior List BL() contains the list of operations and input arguments that are performed in near to implementation steps by entities $S_i$, $A_x$, $A_y$, and $R_k$.

5.1. KDFA Analysis and Verification. In this section, KDFA is analyzed for intra-actor distribution scenario as discussed in Section 4. In this scenario, the key distribution is initiated by the sender node ($S_i$) by transmitting message M$_1$ and M$_2$ to actor node $A_x$ and $A_y$, respectively, and by considering $f = 2$ fragments. $R_k$ receives the frames and calculates the secret key. After sending operation, an update operation is performed to refresh the observer list as shown below.

(i) Concat($K_{F_m, n_i, t_{x+1}}, H_{S,F_m}$) $\rightarrow P_{S,F_m}$

(ii) Encrypt($P_{S,F_m}, K_{S_i, A_y}) \rightarrow C_1$

(iii) Send($A_y$, {ID$_S$, C1}) $\rightarrow M_1$

(iv) Update($n_i, t_{x+1}, K_{F_m}$).

Sender node further unicasts another message to actor node $A_x$ for transmitting the second key fragment using message $M_2$ towards receiver node $R_k$. The steps are similar whereas new parameters are included as shown below.

(i) Hash($h(); K_{F_m, n_i, t_{x+1}}, t_{x+1}) \rightarrow H_{S,F_m}$

(ii) Concat($K_{F_m, R_k, n_i, t_{x+1}}, H_{S,F_m}$) $\rightarrow P_{S,F_m}$

(iii) Encrypt($P_{S,F_m}, K_{S_i, A_y}) \rightarrow C_2$

(iv) Send($A_y$, {ID$_S$, C2}) $\rightarrow M_2$

(v) Update($n_i, t_{x+1}, K_{F_m}$).

Related parameters including messages, ciphers, nonce values, actual key, key fragments, and hashes are saved in possession set at $S_i$ during the key establishment phase as shown below.

POSS($S_i$) = {ID$_S$, K$_{S_i, A_x}$, K$_{S_i, A_y}$, P$_{S,F_m}$, n$_i$, t$_i$, K$_{F_m}$, H$_{S,F_m}$, C$_1$, M$_1$, K$_{F_m, R_k}$, n$_i$, t$_{x+1}$, P$_{S,F_m}$, H$_{S,F_m}$, C$_2$, M$_2$, K$_{S_i, A_y}$}

BEL($S_i$) = {#(ID$_S$), (K$_{S_i, A_x}$), (K$_{S_i, A_y}$)}.

After sending two messages, control is passed to receive operation in appropriate actor node where messages are decrypted to retrieve concatenated message that are further split to get actual parameters. Freshness of message is checked by subtracting timestamp received $t_{i+1}$ from the actor's timestamp $t_{A_y}$, and if the answer is greater than the threshold $\Delta t$ then further calculations are proceeded. Otherwise, message is discarded. MAC is also calculated as MAC({Concat($K_{F_m, n_i, t_{x+1}}$)}) for concatenated parameters including $K_{F_m, n_i}$, $t_i$, and compared to the hash values to ensure integrity protection. These operations, as shown in the following steps, are executed at actor node $A_y$ with similar steps are followed by $A_x$ with different parameters.

(i) Receive($S_i$, {ID$_S$, C1})

(ii) Split({ID$_S$, C1})

(iii) Decrypt($C_1$) to get [K$_{F_m}, R_k, n_i, t_{x+1}, H_{S,F_m}$]

(iv) Check-freshness($t_{A_x} - t_{x+1} \geq \Delta t$) if yes, then abort

(v) MAC({Concat($K_{F_m, n_i, t_{x+1}}$)}) $\rightarrow H^*$

(vi) Check($H_{S,F_m}$, $H^*$).
Actor nodes \(A_x\) and \(A_y\) further send these key fragments towards the receiver node by encrypting with the preestablished key with receiver node \(R_k\).

(i) Concat\((K_{F_m}, n_t, t, H_{S_{F_m}}) \rightarrow P_{A_x}\)

(ii) Encrypt\((\{P_{A_x}\}_{K_{A_x_r_k}}) \rightarrow C_3\)

(iii) Send\((R_k, \{ID_{A_x}, C_3\}) \rightarrow M_3\)

(iv) Update\((MID)\)

(v) Forget\((P_{A_x}, t, K_{F_m}, R_k, C_3, M_3, H^*)\)

Possession set at actor nodes \(A_x\) and \(A_y\) contains the following parameters during execution of the above-mentioned operations as shown below.

\[
\text{POSS}(A_x) = \{ID_{A_x}, K_{A_x,S}, K_{A_x-r_x}, P_{A_x}, R_k, n_t, t, K_{F_m}, C_3, M_3, H^*\}
\]

\[
\text{BEL}(A_x) = \{\#(ID_{A_x}), \#(K_{A_x,S}), \#(K_{A_x-r_x})\}
\]

Possession set values including nonce values \(n_t\), \(n_{t+1}\), time stamps \(t\), \(t_{t+1}\), key fragments \(K_{F_m}\), \(K_{F_{m+1}}\), cipher texts, \(C_1\), \(C_3\), and \(C_5\), hash values \(H_{S_{F_m}}, H^*\), and \(H_{R_k}\), concatenated strings \(P_{S_{F_m}}, P_{S_{F_{m+1}}}, P_{R_k}\), and secure keys are maintained at the receiver node \(R_k\) during the execution of the above-mentioned operations.

\[
\text{POSS}(R_k) = \{ID_{R_k}, R_{K_{R_A}}, A_x, R_{K_{R_A}}, P_{S_{F_m}}, n_t, t, K_{F_m}, H_{S_{F_m}}, H^*, C_1, K_{F_{m+1}}, n_{t+1}, t_{t+1}, P_{S_{F_{m+1}}}, H_{S_{F_{m+1}}}, C_3, H_{R_k}, P_{R_k}, C_5, M_3, K_{R_{K_{R_A}}}\}
\]

\[
\text{BEL}(S_k) = \{\#(ID_{S_k}), \#(K_{S_{A_x}}), \#(K_{S_{A_y}}), \#(K_{S_{R_k}})\}.
\]

After execution of all steps, secure key \(K_{R_{K_{R_A}}}\) is saved in belief set. Unnecessary values are removed by using Forget\((P_{S_{F_m}}, n_t, t, K_{F_m}, H_{S_{F_m}}, H^*, C_1, K_{F_{m+1}}, n_{t+1}, t_{t+1}, P_{S_{F_{m+1}}}, H_{S_{F_{m+1}}}, C_3, H_{R_k}, P_{R_k}, C_5, M_3)\) operation after successful transmission of message from receiver \(R_k\) back to sender \(S_k\).

The sender node \(S_k\) receives the acknowledgement message \(M_5\) encrypted with currently established key \(K_{R_k, R_{K_{R_A}}}\). Node \(S_k\) decrypts the cipher text \(C_5\) to get \([n_t^*, t_k, H_{R_k}]\) and verify the nonce value as shown in the following steps.

(i) Receive\((R_k, \{ID_{R_k}, C_5\})\)

(ii) Split\((\{ID_{R_k}, C_5\})\)

(iii) Decrypt\((\{C_5\}_{K_{R_k, R_{K_{R_A}}}})\) to get \([K_{F_m}, n_t, t, H_{S_{F_m}}]\)

(iv) Check-freshness \((t - t_k) \geq \Delta t\) if yes, then abort

(v) MAC\((\{\text{Concat}(K_{F_m}, n_t, t)\}) \rightarrow H^*\)

(vi) Check\((H_{S_{F_m}}, H^*)\)

(vii) INCR\((m_t, 1)\)

(viii) Compare\((m_t\) equals \(f)\) if no then continue.

If \(m_t\) is equal to \(f\) then \(R_k\) will apply the XOR operation on all the key fragments to calculate the actual key and then sends the acknowledgement message to the sender node \(S_k\) as shown below.

(i) XOR\((K_{F_m}, K_{F_{m+1}}) \rightarrow R_{K_{R_k}}S_k\)

(ii) Hash\((h(\cdot); n_t, t_k) \rightarrow H_{R_k}\)

(iii) Concat\((n_t, t_k, H_{R_k}) \rightarrow P_{R_k}\)

(iv) Encrypt\((\{P_{R_k}\}_{K_{R_k, S_k}}) \rightarrow C_5\)

(v) Send\((S_k, \{ID_{R_k}, C_5\}) \rightarrow M_5\)

(vi) Forget\((P_{S_{F_m}}, n_t, t, K_{F_m}, H_{S_{F_m}}, H^*, C_1, K_{F_{m+1}}, n_{t+1}, t_{t+1}, P_{S_{F_{m+1}}}, H_{S_{F_{m+1}}}, C_3, H_{R_k}, P_{R_k}, C_5, M_5)\).

Possession set at \(S_k\) is refreshed after the completion of key distribution process and receiving the acknowledgement message from the receiver node \(R_k\). Secret key \(K_{R_{K_{R_A}}}\) is also saved in the belief set. \(S_k\) executes the above-mentioned Forget\((P_{S_{F_m}}, n_t, t, K_{F_m}, H_{S_{F_m}}, C_1, M_1, K_{F_{m+1}}, n_{t+1}, t_{t+1}, P_{S_{F_{m+1}}}, H_{S_{F_{m+1}}}, C_3, M_3, H^*)\) operation to clear the unnecessary values from possession set that is shown below.

\[
\text{POSS}(S_k) = \{ID_{S_k}, R_{K_{S_{A_x}}}, K_{S_{A_x}}, R_{K_{S_{A_y}}}, K_{S_{R_k}}\}
\]

\[
\text{BEL}(S_k) = \{\#(ID_{S_k}), \#(K_{S_{A_x}}), \#(K_{S_{A_y}}), \#(K_{S_{R_k}})\}.
\]

### 6. Simulation and Results

For simulation, we have used ns-2.35 on Fedora Core 12. During simulation scenarios, C language is used for providing send and receive functionalities along with packet formats for sensor nodes, actors, and sink. Encryption, decryption, fragmentation, and hash functions are also provided in C. Tcl is used to create nodes, deployment of network field, and linking the nodes. Messages scenarios are also exchanged as per KDP specifications. In WSAN, nodes and actors are randomly deployed in a region of 1200 x 1200 meters where nodes were static and actors can move within the network.
field. We have focused on providing key distribution services and data collection scenario is left for future work. Transmission range of nodes is set to 40 meters with different network sizes from 100 to 1000 nodes. It also includes 5 to 20 powerful actor nodes. Queue type is set to Queue/DropTail/PriQue as shown in Table 3.

### 6.1. Resilience

During interactor communication (IAC) two ordinary nodes from different regions communicate using two actor nodes. The probability that an actor node being compromised in the path can be calculated using (12), where \( N \) is the network size and \( cp \) is the number of nodes captured by an attacker in the network:

\[
P_{NAC} = \left( \frac{N-3}{cp} \right) \left( \frac{N-2}{cp} \right).
\]

The total number of uncompromised nodes \( N - 2 \) means that 2 ordinary nodes are excluded from compromised nodes in network whereas \( N - 3 \) represents the further exclusion of one actor node. The probability that a certain actor node is captured at either sender or receiver side can be calculated using

\[
P_{AC} = 1 - P_{NAC} = \frac{CP}{N-2}.
\]

Considering the scenario of dividing a key into two fragments, we can calculate the probability of capturing exactly \( f = 2 \) actor nodes that are randomly selected for IAC as \( P_{ACf=2} = P_{AC} \times P_{AC} \). We simulated KDF scheme for a network of 1000 ordinary nodes with a transmission range of 40 meters and 50 actor nodes with transmission range of 250 meters.

Results in Figure 7(a) show that if 150 nodes are captured then there are 15% of chances to capture one actor node in the path. In case of fragmentation, there are only 2% for \( f = 2 \) and 0.3% for \( f = 3 \), which are exactly those actor nodes that are randomly selected to transmit the key fragments. Figure 7(b) further evaluates the scenario when more than 60% of the nodes are compromised. If 600 nodes are compromised then there are only 13% of chances for \( f = 4 \), 21.7% for \( f = 3 \), and 36% for \( f = 2 \) scenario that \( f \) Actor nodes are captured. Hence resilience is very much improved due to fragmentation.

### 6.2. Fragmentation Computation Cost

A key fragment size is calculated using KFA as discussed in Section 4.1 where the size of the last fragment varies in case of odd number of fragments. Additional bits are added to last fragment as illustrated in step 7 and that is used to generate (14), where \( L_f \) is the size of the last fragment, \( K_z \) is the key size, and \( f \) is the number of fragments. The last key fragment size depends on the remainder of \( [K_z/f] \) operation:

\[
L_f = K_z + \left\lfloor \frac{K_z}{f} \right\rfloor (1 - f).
\]

Figure 8(a) shows the size of the last fragment by selecting \( f = 2 \), 3 and 4 key fragmentation scenarios for distributing the keys of sizes 16 bits, 32 bits, 64 bits, and 128 bits. In case of \( f = 3 \) fragments, last fragment size is 24 bits whereas initial two fragments are 20 bits for distributing a 64-bit key. These 4 extra bits are produced because \( K_z \) is indivisible by \( f \). Figure 8(b) illustrates the size of the initial key fragments. By considering a 32-bit bit scenario for \( f = 3 \) fragments, the size of the initial two fragments is 10 bits. In case of even value for \( f \), the last fragment size is same as the initial key fragments.

### 6.3. Communication Overhead Analysis

In KDF, the number of messages (\( M_C \)) transmitted during intra- or interactor communication scenarios can be calculated using (15) where \( f \) represents the number of fragments. In this equation, \( M_N \) and \( M_A \) represent the number of messages transmitted by ordinary node and actor, respectively, during intra- or interactor communication scenario without fragmentation:

\[
M_C = (M_N + M_A) \times f.
\]

In case of intra-actor communication, \( M_N = M_A = 1 \) because the actual key was sent in single message like NEKM [20]. By applying fragmentation, the value of \( M_C = 4 \) for \( f = 2 \) fragments using (15). Similarly, in interactor communication scenario, \( M_N = 1 \) and \( M_A = 2 \) and \( M_C = 6 \) for \( f = 2 \) fragments using (15). Our main concern is to calculate the communication cost incurred by ordinary nodes. Therefore, we have ignored the value of \( M_A \). A sender transmits \( (M_N \times f) \) messages to actor nodes, and hence \( (f - 1) \) extra messages are sent by ordinary node due to fragmentation. If 2 key fragments are exchanged then only 2 messages are sent by the ordinary nodes to actors where 1 message is extra as compared to fragmentation less scenarios.

An ordinary node can establish indirect keys with distant sensor nodes in outer circle of its transmission range as illustrated in Figure 9(a). The total communication cost for indirect keying (\( TM_C \)) can be calculated using (16) where \( N_{OC} \) represents the total number of nodes in outer circle:

\[
TM_C = N_{OC} \times (M_N + M_A) \times f.
\]

### Table 3: Simulation parameters.

| Parameters         | Values                        |
|--------------------|-------------------------------|
| Network field      | 1200 \( \times \) 1200 meters |
| Actor Tx radius    | 300 m                         |
| Initial energy at node | 1000 J                        |
| Tx power at node   | 0.819 J                       |
| Receiving power    | 0.049 J                       |
| Physical type      | Wireless physical             |
| Queue type         | Queue/DropTail/PriQue         |
| Antenna type       | Omni antenna                  |
| Max packet in queue| 50                            |
| Routing protocol   | DSDV                          |
| Agent trace        | ON                            |
| Router trace       | ON                            |

International Journal of Distributed Sensor Networks
Figure 7: Probability of an actor node being compromised when less than 40% nodes in the entire network are captured in (a) and more than 60% in (b).

Figure 8: Size in bits of the last fragment in (a) and size of initial fragments in (b) where the number of total fragments is \( f = 2, 3, 4 \).

Figure 9: (a) Effects of key fragmentation on communication overhead. (b) Probability of links between nodes and actors.
Figure 9(a) illustrates that if an ordinary node establishes keys with 6 nodes in its outer circle then 12 messages are exchanged by that node for \( f = 2 \) key fragments. The same number of messages was sent in STSK [10] where an ordinary node was used as intermediary to exchange a single actual key per message without fragmentation. In NEKM [20], only 6 messages were sent to 6 nodes for key distribution using single actor without fragmentation. Message size for exchanging key fragment is 14 bytes that contains 16-bit sender ID, 32-bit key fragment, 16-bit nonce value, 16-bit timestamp, and 32-bit hash value according to (5). Our scheme requires extra 14 bytes to transmit one extra message in case of \( f = 2 \) but such a little amount of overhead ensures the secure key distribution in severe environments even when a large number of nodes or actors are compromised. Fragmentation is only required for indirect environments even when a large number of nodes or actors are compromised. In the proposed scheme, sender transmits the same number of messages to actors whether intra-, interactor, or hybrid communication scenarios is adopted by actors.

If there are \( Q \) actor nodes in the network then \( P_{\text{Actor}} = \left( \frac{N-2}{Q-2} \right) / \left( \frac{N}{Q} \right) \) is the probability that a certain node is an actor. A node being an ordinary node could be identified with a probability \( 1 - P_{\text{Actor}} \). The probability of establishing links with gateway nodes at one end is \( P_{\text{Actor}} \times (1 - P_{\text{Actor}}) \). Figure 9(b) elucidates that in case of 50 actors in a network of 1000 nodes, only 9.5% links are established between actors and nodes, 90% between nodes, and only 0.2% between actors. By referring to these probabilities, we can claim that the chances of intra-actor communication are higher than the interactor communication scenarios.

### 7. Conclusion

Key distribution schemes require agent nodes as intermediaries to establish keys between distant nodes. A compromised actor node can reveal all the keys passing through that node. Therefore, we have proposed a lightweight key fragmentation and assimilation scheme to mitigate the key exposure problem. Rubin logic is applied to model the proposed scheme KDFA and analysis is also performed to evaluate its accuracy of cryptographic operations and transmissions. Simulation results show the effectiveness of the proposed scheme over contemporary schemes in literature. According to results, 75% of communication is compromised in existing schemes whereas the proposed KDFA has the probability of exposing only 13% of communication when 60% of nodes are compromised. In the proposed scheme, sender transmits extra 14 bytes during key distribution process but using powerful actor nodes reduces the overhead for those messages exchanged through actors. In future, the proposed scheme will be analyzed to measure the effects of poor interactor connectivity on data extraction and aggregation.

### Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

### Acknowledgments

The authors would like to extend their sincere appreciation to the Deanship of Scientific Research at King Saud University for funding this research through Research Group Project (RG no. 1435-051).

### References

[1] N.-T. Dinh and Y. Kim, "Potential of information-centric wireless sensor and actor networking," in Proceedings of the International Conference on Computing, Management and Telecommunications (ComManTel ‘13), pp. 163–166, IEEE, January 2013.

[2] W. Abbas, H. Jaleel, and M. Egerstedt, "Energy-efficient data collection in heterogeneous wireless sensor and actor networks," in Proceedings of the 32nd Annual Conference on Decision and Control (CDC ’13), pp. 4164–4169, Firenze, Italy, December 2013.

[3] Binodkumar, B. Malathi, R. Dibalaji, and L. Ramkumar, "Wireless sensor and actor network—security analysis," Journal of Wireless Networking and Communications, vol. 4, no. 1, pp. 1–6, 2014.

[4] F. Xia, Y.-C. Tian, Y. Li, and Y. Sun, "Wireless sensor/actuator network design for mobile control applications," Sensors, vol. 7, no. 10, pp. 2157–2173, 2007.

[5] I. F. Akyildiz and I. H. Kasimoglu, "Wireless sensor and actor networks: research challenges," Ad Hoc Networks, vol. 2, no. 4, pp. 351–367, 2004.

[6] J. Chen, M. Diaz, L. Llopis, B. Rubio, and J. M. Troya, "A survey on quality of service support in wireless sensor and actor networks: requirements and challenges in the context of critical infrastructure protection," Journal of Network and Computer Applications, vol. 34, no. 4, pp. 1225–1239, 2011.

[7] A. Cuevas, M. Ureña, G. de Veciana, R. Cuevas, and N. Crespi, "Dynamic data-centric storage for long-term storage in wireless sensor and actor networks," Wireless Networks, vol. 20, no. 1, pp. 141–153, 2014.

[8] P. Kalyanasundaram, M. Priyadharshini, and T. Gnanasekaran, "The unceasing detection of adjoining nodes, its connectivity, weakness impact on wireless sensor networks and actor networks," vol. 3, no. 1, pp. 1346–1362, 2014.

[9] J. Du, E. Kranakis, and A. Nayak, "Distributed key establishment in disruption tolerant location based social wireless sensor and actor network," in Proceedings of the 9th Annual Communication Networks and Services Research Conference (CNSR ‘11), pp. 109–116, IEEE, Ottawa, Canada, May 2011.

[10] S. K. Addya and A. K. Turuk, "A technique for communication of distance node on key pre-distribution in wireless sensor networks," International Journal on Recent Trends in Engineering & Technology, vol. 4, no. 1, pp. 87–92, 2010.

[11] A. Reddy, "Key management in mobile sensor networks," Journal of Wireless Sensor Networks, vol. 3, no. 6, 2011.

[12] D. Liu, P. Ning, and W. Du, "Group-based key predistribution for wireless sensor networks," ACM Transactions on Sensor Networks, vol. 4, no. 2, article 11, 2008.
[13] A. D. Rubin and P. Honeyman, "Nonmonotonic cryptographic protocols," in *Proceedings of the Computer Security Foundations Workshop VII (CSFW ’94)*, pp. 100–116, June 1994.

[14] D. Huang and D. Medhi, "Secure pairwise key establishment in large-scale sensor networks: an area partitioning and multi-group key predistribution approach," *ACM Transactions on Sensor Networks*, vol. 3, no. 3, article 16, 2007.

[15] L. Zhou, J. Ni, and C. V. Ravishankar, "Efficient key establishment for group-based wireless sensor deployments," in *Proceedings of the ACM Workshop on Wireless Security (WiSe ’05)*, pp. 1–10, September 2005.

[16] P. Kumar, A. J. Choudhury, M. Sain, S.-G. Lee, and H.-J. Lee, "RUASN: a robust user authentication framework for wireless sensor networks," *Sensors*, vol. 11, no. 5, pp. 5020–5046, 2011.

[17] N. Mehallegue, A. Bouridane, and E. Garcia, "Efficient path key establishment for wireless sensor networks," *EURSIP Journal on Wireless Communications and Networking*, vol. 2008, Article ID 456703, 2008.

[18] A. Alfadhly, U. Baroudi, and M. Younis, "Optimal node repositioning for tolerating node failure in wireless sensor actor network," in *Proceedings of the 25th Queen’s Biennial Symposium on Communications (QBSC ’10)*, pp. 67–71, May 2010.

[19] F. G. Mármol, C. Sorge, O. Ugus, and G. M. Pérez, "WSAN-Rep, WSAN reputation-based selection in open environments," *Wireless Personal Communications*, vol. 68, no. 3, pp. 921–937, 2013.

[20] Y. Lee and S. Lee, "A new efficient key management protocol for wireless sensor and actor networks," *International Journal of Computer Science and Information Security*, vol. 6, no. 2, pp. 15–22, 2009.

[21] Y. Zhou and Y. Fang, "A two-layer key establishment scheme for wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 9, pp. 1009–1020, 2007.

[22] A. Ghafoor, M. Sher, M. Imran, and K. Saleem, "A lightweight key freshness scheme for wireless sensor networks," in *Proceedings of the 12th International Conference on Information Technology: Next Generation (ITNG ’15)*, Las Vegas, Nev, USA, April 2015.

[23] S. S. Kashi and M. Sharifi, "Connectivity weakness impacts on coordination in wireless sensor and actor networks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 145–166, 2012.

[24] S. Acharya and C. R. Tripathy, "Inter-actor connectivity restoration in wireless sensor actor networks: an overview," in *ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India—Vol I*, vol. 248 of *Advances in Intelligent Systems and Computing*, pp. 351–360, Springer, 2014.

[25] M. Imran and N. A. Zafar, "Formal specification and validation of a hybrid connectivity restoration algorithm for wireless sensor and actor networks," *Sensors*, vol. 12, no. 9, pp. 11754–11781, 2012.

[26] S.-G. Wang, X.-F. Mao, S.-J. Tang, X.-Y. Li, J.-Z. Zhao, and G.-J. Dai, "On movement-assisted connectivity restoration in wireless sensor and actor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 4, pp. 687–694, 2011.

[27] H. Salarian, K.-W. Chin, and F. Naghdy, "Coordination in wireless sensor-actuator networks: a survey," *Journal of Parallel and Distributed Computing*, vol. 72, no. 7, pp. 856–867, 2012.

[28] F. Kazemeyni, O. Owe, E. B. Johnsen, and I. Balasingham, "Learning-based routing in mobile wireless sensor networks: applying formal modeling and analysis," in *Proceedings of the 2013 14th International Conference on Information Reuse and Integration*, pp. 504–511, San Francisco, Calif, USA, August 2013.
