On the Formal Verification of a Cluster Based Key Management Protocol for Wireless Sensor Networks

Reiner Dojen, Fan Zhang, Tom Coffey
Data Communications Security Laboratory,
Department of Electronic & Computer Engineering
University of Limerick, Ireland
reiner.dojen@ul.ie, fan.zhang@ul.ie, tom.coffey@ul.ie

Abstract

Security is a primary problem in wireless electronic communications. However, the limitations of Wireless Sensor Networks (small size, low battery capacity and exposed communication media) often turn the focus to power consumption rather than security. However, insufficient security protection can leave a WSN application open to attack, rendering the supplied service not trustworthy or unavailable. This paper presents a formal verification of a cluster based key management protocol for Wireless Sensor Networks. A number of weaknesses are identified in the verified protocol that can be exploited by an intruder. These weaknesses allow various attacks, such as impersonating the base station to the network, impersonating a cluster head to sensor nodes or to disrupt the network configuration. A detailed discussion on these weaknesses is presented and modifications to the protocol are proposed to fix these weaknesses. Successful verification of the proposed fixed protocol demonstrates its correctness.

1. Introduction

Wireless Sensor Networks (WSN) are ad hoc networks composed of small network nodes that have the characteristics of low power capacity, small resource storage and relatively low computation capability [1]. One or more base stations (BS) are used to interconnect all nodes and manage communication with the outside of the wireless sensor network. While the BS often has more resources available than generic sensor nodes, the relatively small-scale size of sensor nodes indirectly decides its limited power supply and data storage [9]. Further, the dynamic network topology restricts protection of the infrastructure [7].

To accommodate these resource limitations, many WSN system designers tend to focus on power consumption and neglect security issues [12]. However, insufficient security protection can leave a WSN application open to attack, rendering the supplied service not trustworthy or unavailable.

On the other hand, the design of an effective security protocol is an error prone process. Formal verification of a security protocol in is an essential part of the design process [3], as it:

- provides a systematic way to detect design flaws
- identifies the exact cryptographic properties a protocol aims to satisfy
- identifies the assumptions and the environment under which these properties hold
- removes ambiguity in the specifications of the protocol

Techniques such as state machines/model checkers or modal logics can be used to formally verify security protocols [4]. Both techniques have been shown to be effective in detecting protocol weaknesses [2, 3, 10, and 11].

This paper presents the formal verification of a cluster based key management protocol used for WSN applications. The CBKM protocol by Shen et al. is reviewed and subjected to formal verification using a modal logic. Several weaknesses are detected in the protocol that can be exploited by an attacker. These weaknesses are discussed and modifications to the protocol are proposed to fix these weaknesses. The fixed protocol is verified successfully, demonstrating the correctness of the fixed protocol.

2. A Cluster Based Key Management Protocol (CBKM) for WSN

L. Shen et al. present a Cluster Based Key Management protocol (CBKM) for WSN in [8]. The protocol is designed to be a securely dynamic clustering protocol with a flexible key management capability. The protocol utilizes the Exclusion Basis System (EBS) algorithm [6] to manage group encryption keys and the LEACH protocol [5] to elect

new nodes as cluster heads (CH). The CBKM protocol is based on the following assumptions:

- Each sensor node possesses a unique ID and a secret key initially. The key is known only by it and the BS.
- Each cluster has a unique ID and cluster key *Kc*. *Kc* is known by every member node and the CH of the cluster. Member nodes can only communicate with each other through the CH.
- Each sensor nodes possesses a global initiate key *Kinit* prior to deployment that is used for the first cluster generating process. Kinit is discarded once the first cluster setup is completed.
- CHs are ordinary sensor nodes with the responsibility of integrating their members' data together and forwarding them to the BS.
- The base station has more power and can communicate with all CHs in the network with single hop. The base station processes all sensor data and is in charge of managing keys for the network. In addition, the base station knows the geographic position of all sensor nodes in the network.

EBS is a combinatorial formulation of key management, where each node is assigned a number of keys out of a pool of keys. If one or more nodes are captured, new keys are distributed using only keys that are not known to the captured nodes.

The LEACH (Low-Energy Adaptive Clustering Hierarchy) protocol is a clustering protocol that randomly elects node to CHs in order to distribute the energy load evenly among the sensor nodes in the network [5]. During this process, nodes self-organize into multiple clusters.

The CBKM protocol is composed of two sub-protocols: Firstly, the initial clustering protocol, in which the nodes of the WSN are organised into clusters and the cluster heads are elected. Secondly, the Cluster Role Changing Protocol, which re-elects new cluster heads allows nodes to change to another cluster. The following notations are used throughout this paper to describe protocols:

- *BS: Base station*
- *A, B: Sensor nodes*
- *I, J: CHs*
- *IDX: ID of sensor node/CH X*
- *IDCY: ID of the cluster with header Y*
- *Kinit: Initial key for first cluster establishment*
- *Kx: Node X's key shared with the BS*
- *Kcy: Cluster Y's cluster key*
- *Kebs: EBS management key set*
- *{M}K: The encryption of message M using key K*
- *, : Concatenation of data and messages*
- *→: Unicast*
- *⇒: Broadcast*
- *dataHELLO: HELLO message*
- *dataACK: Acknowledgement message*

2.1 The Initial Clustering Protocol

The initial clustering protocol (as shown in Figure 1) executes immediately after system initialisation. Some sensor nodes (node I in Figure 1) elect themselves to be the CHs, based on the algorithm presented in LEACH [5]. These nodes broadcast a HELLO message that includes their identity. This message is encrypted with the key Kinit. The sensor nodes (A) in the broadcasting range of the elected cluster heads receive (potentially multiple) HELLO messages (dataHELLO) and decide to which cluster they will belong. Node A sends an acknowledge message (dataACK), consisting of A's identity and the ACK flag encrypted with Kinit, to the selected cluster's head (I). Each cluster head will wait for a defined time period to receive acknowledgement messages. Afterwards, the cluster heads send a message to the BS informing it about the cluster configuration. This messages contains the ID's of all nodes in the cluster and is encrypted with the key I shares with BS (Figure 1 assumes that nodes A and B are in I's cluster). For each cluster the BS generates an ID, a cluster key Kc1 and an EBS management key set Kebs. The BS then sends a message to each node (X) containing the generated information for the node's cluster. These messages are encrypted with the corresponding key shared between X and the BS. Now the clusters are formed, and the global key Kinit is discarded.

1. I => : {dataHELLO}Kinit
2. A→I: {IDA, dataACK}Kinit
3. I→BS: {IDA, IDB}Ki
4. BS→ I: { IDC1, Kc1, Kebs}Ki
5. BS→A: {IDC1, Kc1, Kebs}Ka

Figure 1: The initial clustering protocol

2.2 The Cluster Role Changing Protocol

The Cluster Role Changing Protocol allows re-electing cluster heads and cluster membership of nodes to distribute increased power consumption due to cluster head duties among all nodes. This protocol is operated periodically, as one sensor node cannot act as the cluster head for too long, otherwise its battery will be used up soon. Figure 2 outlines the Cluster Role Changing Protocol for a cluster with current head I, new cluster head J and a node A that wants to change from H's cluster to J's cluster. As in the initial

clustering protocol, nodes use the LEACH algorithm to elect new cluster heads. The new cluster head J sends an application message via the current cluster head I to the BS. This message is encrypted with the key shared between J and the BS. The BS replies (via the current cluster head I), by sending the keys of three geographically neighbouring clusters to node J, which then uses the keys to encrypt and broadcast the HELLO message to the sensor nodes in its cluster and the nearby clusters (Figure 2 assumes that J has three neighbouring clusters, with the cluster key Kc1, Kc2, and Kc3 respectively). As nodes that want to become members of J's cluster do not share a secret key with J, they send an acknowledge message via their current cluster head to the BS. These messages are encrypted with respective key shared between the node and the BS. When all acknowledgements are received, the BS generates new cluster keys and new EBS management key sets Kebs. Similar to the initial clustering protocols, the BS then sends a message to the nodes containing the generated information for the node's cluster. Finally, the BS sends a message to the new cluster head that contains the IDs of all nodes in the corresponding cluster, encrypted with the key shared between the cluster head and the BS.

1. J→I→BS: {dataAPP}Kj
2. BS→I→J: {Kc1, Kc2, Kc3}Kj
3. J => : {dataHELLO}Kc1, {dataHELLO}Kc2, {dataHELLO}Kc3
4. A→L→BS: {IDA, dataACK}Ka
5. BS→L→A: {dataIDC4, Kc4, Kebs}Ka
6. BS→I→J: {IDA, IDB…, dataIDC4, Kc4, Kebs}Kj

Figure 2: The cluster role changing protocol

3. Formal Verification of CBKM Protocol

This section uses the automated logic-based CDVT tool [13,14] to verify the correctness of the CBKM protocol. The CDVT verification engine is an automated system that implements a modal logic of knowledge and belief using Layered Proving Trees. The implemented logic can analyze the evolution of both knowledge and belief during a protocol execution and is therefore useful in addressing issues of both security and trust. For details on the input language of the CDVT tool see [14].

In order to apply the CDVT tool, the protocol must be formalised, i.e. translated into the formal language of the tool. A formalised protocol consists of three components:

- Initial assumptions (that hold before protocol start)
- Protocol steps (the exchanged messages)
- Protocol goals (conditions that are expected to hold)

3.1. Verification of the Initial Clustering Protocol

Verification of the initial clustering protocol requires only modelling the BS, one node that elects to become a cluster head and one node that chooses to become a member of the cluster. The same configuration has been used to introduce this protocol in Figure 1. Given these restrictions, the following specifies the initial assumptions of the initial clustering protocol:

A1: A possess at[0] Ka;
A2: I possess at[0] Ki;
A3: BS possess at[0] Ka;
A4: BS possess at[0] Ki;
A5: A possess at[0] Kinit;
A6: I possess at[0] Kinit;
A7: A know at[0] I possess at[0] Kinit;
A8: I know at[0] A possess at[0] Kinit;
A9: A know at[0] BS possess at[0] Ka;
A10: A know at[0] BS possess at[0] Ka;
A11: I know at[0] BS possess at[0] Ki;
A12: BS know at[0] I possess at[0] Ki;

The initial assumptions A1 to A6 model the initial possessions of shared keys: node A shares key Ka with BS, node I shares key Ki with BS and nodes A and I share key Kinit. Assumptions A7 to A12 model the knowledge of sharing the keys with the corresponding nodes. The protocol steps of the initial clustering protocol are formalised as follows:

S1: A receive at[1] {I, dataHELLO}Kinit;
S2: I receive at[2] {A, dataACK}Kinit;
S3: BS receive at[3] {A}Ki;
S4: I receive at[4] {dataIDC1, Kc1, Kebs}Ki;
S5: A receive at[5] {dataIDC1, Kc1, Kebs}Ka;

The initial clustering protocol aims to fulfil the following goals:

G1: I possess at[4] Kc1;
G2: A possess at[5] Kc1;
G3: I possess at[4] Kebs;
G4: A possess at[5] Kebs;
G5: I know at[4] BS send at[4] {dataIDC1, Kc1, Kebs}Ki;
G6: I know at[4] NOT(Zero send at[0] {dataIDC1, Kc1, Kebs}Ki);
G7: A know at[5] BS send at[5] {dataIDC1, Kc1, Kebs}Ka;
G8: A know at[5] NOT(Zero send at[0] {dataIDC1, Kc1, Kebs}Ka);

The goals G1 to G4 specify that the cluster head I and node A in the cluster possess the cluster key Kc1 and the EBS key management set Kebs. Goal 5 specifies that the cluster head can confirm that message S4 has been send by the BS and Goal 6 states that the cluster head (I) can establish that message S4 is not a replay from a previous protocol run. Goals 7 and 8 detail the corresponding knowledge of sensor nodes about message S5.

The specifications of the initial assumptions, the protocol steps and the protocol goals are now analysed with the automated verification tool. The obtained results are presented in Figure 3.

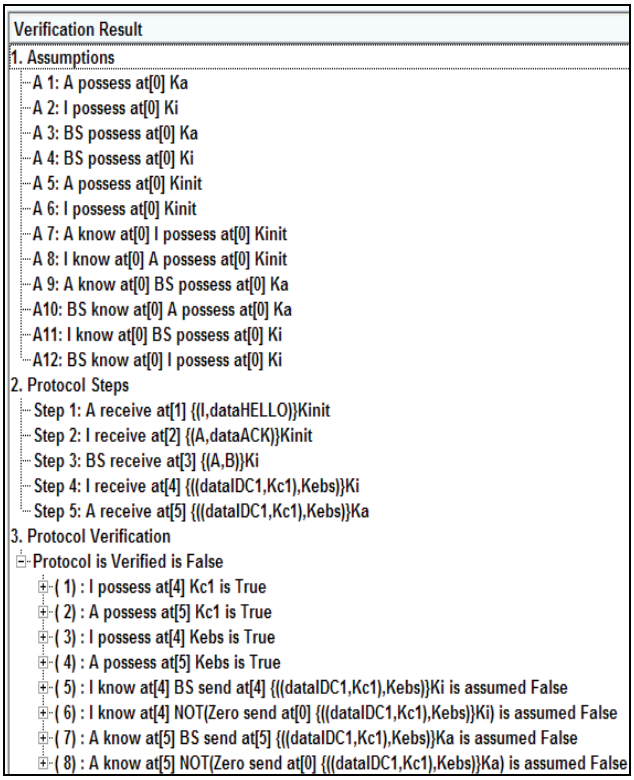

Figure 3: Initial clustering protocol verification result

3.2. Analysis of Initial Clustering Protocol Verification Results

The verification of the initial clustering protocol fails. As shown in Figure 3, the goals G5 to G8 are not achieved. A thorough analysis of the verification of the failed goals reveals design flaws in the protocol. For example, Figure 4 presents a detailed view of the attempted verification of goal G5. For example, the statement with number 205 indicate that the cluster head I is not able to determine the freshness of the data components in message S4. This means, that I is not able to distinguish the received message from a potentially replayed message.

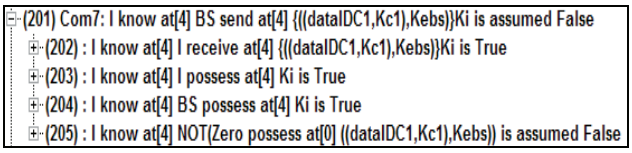

Figure 4: Detailed verification result of G5

Analysis of the other failed goals reveals that the failure of goals G5 to G8 stem from the following two weaknesses:

- Cluster head I cannot establish freshness of S4
- Sensor node cannot establish freshness of S5

As a result, an intruder may replay messages S4 and S5 from previous protocol runs. If the keys from the previous protocol run are compromised, this allows an intruder to masquerade as the BS to all cluster heads and sensor nodes in the WSN.

3.3. Verification of the Cluster Role Changing Protocol

For the verification of the Cluster Role Changing Protocol we adopt the same configuration as has been used in Section 2.2: It is assumed that there are currently 3 clusters with cluster keys Kc1, Kc2 and Kc3, respectively. Node I is currently head of cluster 1 and node H is head of cluster 2. As forwarding message 1 by cluster head I and message 4 by cluster head H have no security impact, this verification assumes these messages are sent directly to the BS. Thus, presence of nodes I and H is not required in this specification. Node J is part of cluster 1 and elects itself to become the new cluster head and node A is currently part of cluster 2 and chooses to become a member of the new cluster 4. Given this configuration, the following formalises the initial assumption of the Cluster Role Changing Protocol:

A1: A possess at[0] Ka;
A2: A possess at[0] Kc2;
A3: J possess at[0] Kj;
A4: BS possess at[0] Ka;
A5: BS possess at[0] Kj;
A6: A know at[0] BS possess at[0] Ka;
A7: BS know at[0] A possess at[0] Ka;
A8: J know at[0] BS possess at[0] Kj;
A9: BS know at[0] J possess at[0] Kj;

These initial assumptions correspond directly to the ones presented for the initial clustering protocol: Assumptions A1 to A5 represent possession of the shared keys and assumptions A6 to A9 model knowledge of sharing these keys with the

corresponding principals. The protocol steps of the role changing protocol are formalised as follows:

S1: BS receive at[1] {J, dataAPP}Kj;
S2: J receive at[2] {Kc1, Kc2, Kc3}Kj;
S3: A receive at[3] {J, dataHELLO}Kc1, {J, dataHELLO}Kc2, {J, dataHELLO}Kc3;
S4: BS receive at[4] {A, dataACK}Ka;
S5: A receive at[5] {dataIDC4, Kc4, Kebs}Ka;
S6: J receive at[6] {A, dataIDC4, Kc4, Kebs}Kj;

The role changing protocol has the following goals:

G1: J possess at[2] Kc1,Kc2,Kc3;
G2: J know at[2] BS send at[2] {Kc1, Kc2, Kc3}Kj;
G3: J know at[2] NOT(Zero send at[0] {Kc1, Kc2, Kc3}Kj);
G4: A possess at[5] Kc4;
G5: A possess at[5] Kebs;
G6: A know at[5] BS send at[5] {dataIDC4, Kc4, Kebs}Ka;
G7: A know at[5] NOT(Zero send at[0] {dataIDC4, Kc4, Kebs}Ka);
G8: J possess at[6] Kc4;
G9: J possess at[6] Kebs;
G10: J know at[6] BS send at[6] {A, dataIDC4, Kc4, Kebs}Kj;
G11: J know at[6] NOT(Zero send at[0] {A, dataIDC4, Kc4, Kebs}Kj);

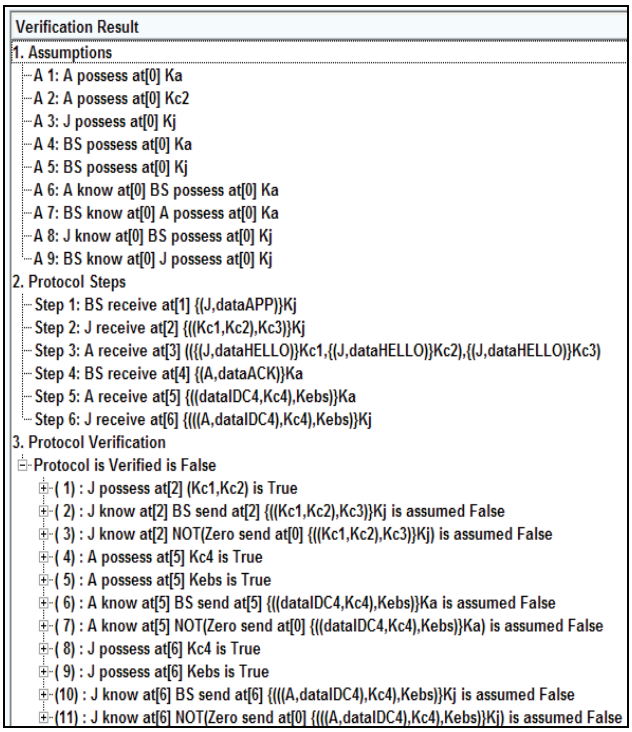

Figure 5: Cluster role changing protocol verification result

G1 states that J receives the cluster keys of the neighbouring clusters. G2 and G3 indicate that J can confirm that message S2 was sent by BS and was sent during the current protocol run. The goals G4 and G5 model A's possession of the new cluster key Kc4 and the EBS management key set Kebs, while goals G6 states that A can confirm BS as source of message S5. G7 indicates that A is able to establish freshness of message S5. Goals G8 to G11 model the corresponding knowledge of J regarding Kc4, Kebs and message S6.

The specifications of the initial assumptions, the protocol steps and the protocol goals are now analysed with the automated verification tool. The obtained results are presented in Figure 5.

3.4. Analysis of Cluster Role Changing Protocol Verification Results

Verification of the Cluster Role Changing Protocol fails. The goals G2, G3, G6, G7, G10 and G11 are not satisfied. The verification reveals that the receiving nodes cannot establish freshness of the corresponding messages and, consequently, cannot establish the source of these messages. In particular:

- New cluster head J cannot establish source of message S2. Consequently, an intruder can inject a bogus message with incorrect keys. As a result, no nodes will be able to join the new cluster.
- Node A is unable to determine freshness of message S5. An intruder can replay an old message to the node that contains compromised keys. As a result, the intruder can impersonate the cluster head to A.
- Cluster head J cannot determine freshness of message S6. An intruder can replay an old message to the cluster head that contains compromised keys. As a result, the intruder can impersonate nodes to the cluster head J.

4. Fixing the Cluster Based Key Management Protocol

The previous section revealed and discussed weaknesses in both sub-protocols of the Cluster Based Key Management Protocol. Here we propose modifications to the protocol to fix the discussed weaknesses. The fixed protocols are also formally verified. The presented formal verifications in this section assume the same WSN configurations as have been used to verify the original protocols.

4.1. Fixing the Initial Clustering Protocol

Formal verification of the Initial Clustering Protocol revealed that freshness of messages 4 and 5 cannot be established by the receiving nodes. Thus, an intruder may replay messages from previous protocol runs that contain compromised keys. To remove this weakness, the cluster head and the sensor nodes can add nonces (once off random numbers) to the protocol steps: In step 2 sensor nodes send their nonces (Na) to the cluster head and in step 3 the cluster head forwards this nonce together with its own nonce (Ni) to the BS. Messages 4 and 5 send by the BS to the cluster head and the sensor node now contain a component that the recipient can identify as fresh. Consequently, the receiving nodes can establish freshness of the entire message. Hence, an intruder is no longer able to replay old messages. The protocol steps of the fixed Initial Clustering Protocol are shown in Figure 6.

1. I => : {I, dataHELLO}Kinit
2. A→I: {A, {Na}Ka, dataACK}Kinit
3. I→BS: {A, B, {Na}Ka, {Nb}Kb, Ni}Ki
4. BS→ I: {BS, Ni, IDC1, Kc1, Kebs}Ki
5. BS→A: {BS, Na, IDC1, Kc1, Kebs}Ka

Figure 6: Fixed initial clustering protocol

4.2. Verification of the Fixed Initial Clustering Protocol

The formalised initial assumptions for the fixed Initial Clustering Protocol are essentially the same as for the original protocol. However, the following additional assumptions are required to model the nodes knowledge about their nonces:

A13: I possess at[0] Ni;
A14: A possess at[0] Na;
A15: I know at[0] NOT(Zero possess at[0] Ni);
A16: A know at[0] NOT(Zero possess at[0] Na);

Formalisation of the protocol steps must address the changes made to the protocol as follows:

S1: A receive at[1] {I, dataHELLO}Kinit;
S2: I receive at[2] {A, {Na}Ka, dataACK}Kinit;
S3: BS receive at[3] {A, B, {Na}Ka, {Nb}Kb, Ni}Ki;
S4: I receive at[4] {dataIDC1, Kc1, Kebs, Ni}Ki;
S5: A receive at[5] {dataIDC1, Kc1, Kebs, Na}Ka;

The protocol goals are also essentially the same as for the original protocol. However, due to the changes in the protocol steps some of the goals change slightly as presented below:

G1: I possess at[4] Kc1;
G2: A possess at[5] Kc1;
G3: I possess at[4] Kebs;
G4: A possess at[5] Kebs;
G5: I know at[4] BS send at[4] {dataIDC1, Kc1, Kebs, Ni}Ki;
G6: I know at[4] NOT(Zero send at[0] {dataIDC1, Kc1, Kebs, Ni}Ki);
G7: A know at[5] BS send at[5] {dataIDC1, Kc1, Kebs, Na}Ka;
G8: A know at[5] NOT(Zero send at[0] {dataIDC1, Kc1, Kebs, Na}Ka);

The verification of the fixed Initial Clustering Protocol succeeds with the result shown in Figure 7.

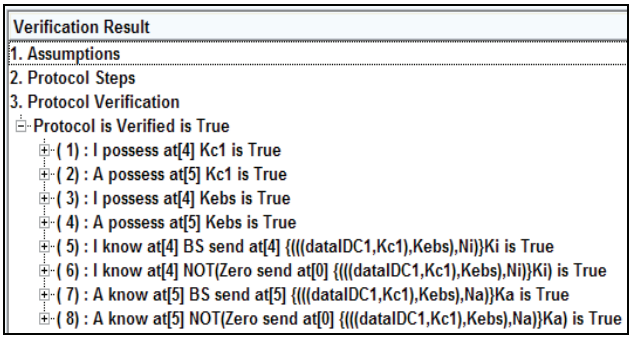

Figure 7: Fixed initial clustering protocol verification result

4.3. Fixing the Cluster Role Changing Protocol

The presented verification of the Cluster Role Changing Protocol in Section 4 exposed three weaknesses that can be exploited by an intruder. Adding nonces to several messages can remove theses weaknesses.Including a nonce (Nj1) created by J into message 1 that can be send back by the BS in message 2 removes the first weakness. As J now can identify a component of message 2 as fresh it can determine the entire message as fresh. Thus, with the added assumption that the key Kj has not been compromised, J can establish that message 2 has been sent by the BS during the current protocol run. Hence, an intruder cannot replay old messages. If node A adds a nonce (Na) to message S4, then the BS can send this nonce back to A in message S5. Thus, node A can establish freshness of message S5 and the presented replay attack is no longer possible.

Additionally, a nonce (Nj2) added by the new cluster head (J) to the broadcasted HELLO message can be forwarded by nodes in the ACKNOWLEDGE message S4 to the BS. The BS will include this nonce in message S6, allowing the new cluster head to identify the freshness of this message. Thus, an intruder cannot replay this message from previous

protocol runs. The resulting fixed Cluster Role Changing Protocol is presented in Figure 8.

1. J→I→BS: {J, Nj1, dataAPP}Kj
2. BS→I→J: {Nj1, Kc1, Kc2, Kc3}Kj
3. J => : {Nj2, J, dataHELLO}Kc1,
   {Nj2, J, dataHELLO}Kc2,
   {Nj2, J, dataHELLO}Kc3
4. A→L→BS: {A, Nj2, Na, dataACK}Ka
5. BS→L→A: {Na, dataIDC4, Kc4, Kebs}Ka
6. BS→I→J: {Nj2, A, dataIDC4, Kc4, Kebs}Kj

Figure 8: Fixed cluster role changing protocol

4.4. Verification of the Fixed Cluster Role Changing Protocol

The formalised initial assumptions for the fixed Cluster Role Changing Protocol are similar to the original protocol. Only the following additional assumptions are required to model the knowledge about nonces:

A10: A possess at[0] Na;
A11: A know at[0] NOT(Zero possess at[0] Na);
A12: J possess at[0] Nj1;
A13: J know at[0] NOT(Zero possess at[0] Nj1);
A14: J possess at[0] Nj2;
A15: J know at[0] NOT(Zero possess at[0] Nj2);

Formalisation of the protocol steps must address the changes made to the protocol. The following formalised steps are obtained:

S1: BS receive at[1] {J, Nj1, dataAPP}Kj;
S2: J receive at[2] {Nj1, Kc1, Kc2, Kc3}Kj;
S3: A receive at[3] {Nj2, J, dataHELLO}Kc1, {Nj2, J, dataHELLO}Kc2, {Nj2, J, dataHELLO}Kc3;
S4: BS receive at[4] {A, Nj2, Na, dataACK}Ka;
S5: A receive at[5] {Na, dataIDC4, Kc4, Kebs}Ka;
S6: J receive at[6] {Nj2, A, dataIDC4, Kc4, Kebs}Kj;

The protocol goals are equivalent to the goals of the original protocol. However, the changes to the protocol steps must be reflected in the goals. This results in the following goals:

G1: J possess at[2] Kc1,Kc2,Kc3;
G2: J know at[2] BS send at[2] {Nj1, Kc1, Kc2, Kc3}Kj;
G3: J know at[2] NOT(Zero send at[0] {Nj1, Kc1, Kc2, Kc3}Kj);
G4: A possess at[5] Kc4;
G5: A possess at[5] Kebs;
G6: A know at[5] BS send at[5] {Na, dataIDC4, Kc4, Kebs}Ka;
G7: A know at[5] NOT(Zero send at[0] {Na, dataIDC4, Kc4, Kebs}Ka);
G8: J possess at[6] Kc4;
G9: J possess at[6] Kebs;
G10: J know at[6] BS send at[6] {Nj2, A, dataIDC4, Kc4, Kebs}Kj;
G11: J know at[6] NOT(Zero send at[0] {Nj2, A, dataIDC4, Kc4, Kebs}Kj);

Figure 9 presents the verification results of the fixed Cluster Role Changing Protocol. The successful verification succeeds of the protocol demonstrates its correctness.

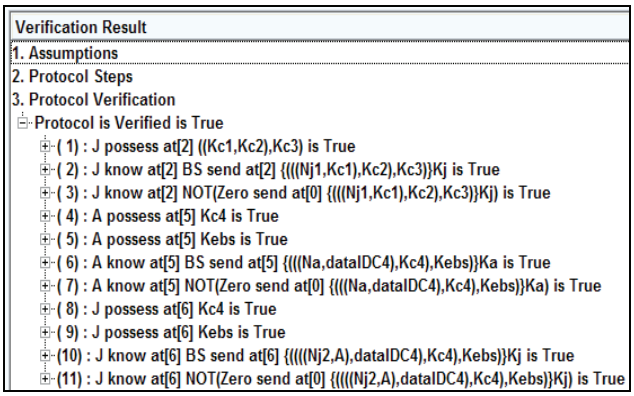

Figure 9: Fixed cluster role changing protocol verification results

5. Conclusion

This paper discussed the security of the Cluster Based Key Management protocol for Wireless Sensor Networks. Formal verification of the sub-protocols of the CBKM revealed a number of weaknesses.

Firstly, freshness weaknesses in the Initial Clustering Protocol allow an attacker to impersonate the BS to cluster heads and sensor nodes, if an old message with compromised keys has been recorded. This allows an attacker to access the entire data recorded by the WSN.

Secondly, freshness weaknesses in the Cluster Role Changing Protocol allow an intruder to distribute bogus keys to cluster heads, which will disallow any nodes to join the cluster. Consequently, the organisation of the network is disrupted.

Thirdly, freshness weaknesses in the Cluster Role Changing Protocol allow an intruder to impersonate a cluster head to sensor nodes and, thus, receiving the data collected by the sensor nodes in the attacked cluster. Alternatively, the intruder can impersonate a sensor node to the cluster head. This allows an intruder

to send incorrect data to the BS via the cluster head and, thus, making the service of the WSN application unreliable.

Corrections were proposed to both sub-protocols to remove the detected weaknesses. Successful formal verification of the fixed protocols demonstrated their correctness.

6. Acknowledgements

This work was funded by the Irish Research Council for Science, Engineering and Technology (IRCSET Embark Initiative)

7. References

[1] K. Romer, and F. Mattern, "The design space of wireless sensor networks", *Wireless Communications*, IEEE, Volume 6 (11), Dec 2004, pp. 54-61.

[2] G. Bella, F. Massacci, L.C. Paulson, and P. Tramontano, "Formal verification of cardholder registration in SET", *European Symposium on Research in Computer Security Computer*, Toulouse, France, Oct 2000, pp. 159–174

[3] T. Coffey, and R. Dojen, and T. Flanagan, "Formal verification: an imperative step in the design of security protocols", *Computer Networks*, Volume 43(5), Dec 2003, pp. 601-618.

[4] R. Dojen, I. Lasc, T. Coffey, and R. Gyorodi, "Verifying a Key Distribution and Authentication Protocol Using a Logic-Based Proving Engine", *7th International Conference on Renewable Sources and Environmental Electro-Technologies*, 29-30 May 2008.

[5] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient communication protocol for wireless microsensor networks", *Proceeding of the 33rd Annual Hawaii Int'l Conference on System Sciences*, IEEE Computer Society, 2000, pp. 3005-3014.

[6] E. Khabiri, and S. Bettayeb, "Efficient Algorithms for Secure Multicast key Management", *Proceedings the 31st IEEE Conference on Local Computer Networks*, Nov. 2006, pp. 787-792.

[7] L. B. Oliveira, H. C. Wong, M. Bern, R. Dahab, and A. A. F. Loureiro, "SecLEACH - A Random Key Distribution Solution for Securing Clustered Sensor Networks". *Proceeding of Fifth IEEE International Symposium on Network Computing and Applications (NCA'06)*, 2006, pp. 135-154.

[8] L. Shen; H. Feng, Y. Qiu, and H. Ding, "A New Kind of Cluster-based Key Management Protocol in Wireless Sensor Network", *Proceedings of the IEEE Conference of Networking*, *Sensing and Control*, 2008.

[9] E. Shi, and A. Perrig, "Designing secure sensor networks", *IEEE Wireless Communications*, Volume 11 (6), 2004, pp. 38-43.

[10] M. Ventuneac, R. Dojen and T. Coffey, "Automated Verification of Wireless Security Protocols using Layered Proving Trees", *WSEAS Transactions on Communications,* Volume 5(2), 2006, pp. 252-258

[11] Y. Zhang and V. Varadharajan, "A logic for modelling the dynamics of beliefs in cryptographic protocols", *Australasian Computer Science Conference*, Gold Coast, Queensland, Australia, February 2001, pp. 215-222.

[12] C. Karlof and D. Wagner, Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures, *IEEE International Workshop on Sensor Network Protocols and Applications*, 2002.

[13] R. Dojen, and T. Coffey, "The concept of layered proving trees and its application to the automation of security protocol verification", *ACM Transactions on Information and System Security (TISSEC)*, ACM, 2005, pp. 287-311.

[14] R. Dojen, I. Lasc, and T. Coffey, "Establishing and Fixing a Freshness Flaw in a Key-Distribution and Authentication Protocol", *2008 IEEE International Conference on Intelligent Computer Communication and Processing*, Cluj-Napoca, Romania, August 28-30, 2008