

# Model Checking Wireless Sensor Network Security Protocols: TinySec + LEAP \*

Llanos Tobarra, Diego Cazorla, Fernando Cuartero, Gregorio Díaz, and Emilia Cambronero

Escuela Politécnica Superior de Albacete  
Universidad de Castilla-La Mancha. 02071 Albacete, Spain.  
{mtobarra, dcazorla, fernando, gregorio,emicp}@dsi.uclm.es

**Abstract.** In this paper, a formal analysis of security protocols in the field of wireless sensor networks is presented. Two complementary protocols, TinySec and LEAP, are modelled using the high-level formal language HLPSL, and verified using the model checking tool Avispa, where two main security properties are checked: authenticity and confidentiality of messages. As a result of this analysis, two attacks have been found: a man-in-the-middle- attack and a type flaw attack. In both cases confidentiality is compromised and an intruder may obtain confidential data from a node in the network. Two solutions to these attacks are proposed in the paper.

**Keywords:** Wireless sensor, model checking, security protocols, avispa toolbox

## 1 Introduction

Security has become a challenge in wireless sensor networks. Low capabilities of devices, in terms of computational power and energy consumption, make difficult using traditional security protocols.

Two main problems related to security protocols arise. Firstly, the overload that security protocols introduce in messages should be reduced at a minimum; every bit the sensor sends consumes energy and, consequently, reduces the life of the device. Secondly, low computational power implies that special cryptographic algorithms that require less powerful processors need to be used. The combination of both problems lead us to a situation where new approaches or solutions to security protocols need to be considered. These new approaches take into account basically two main goals: reduce the overhead that protocol imposes to messages, and provide reasonable protection while limiting use of resources.

---

\* This work has been supported by the Spanish government with the project "Application of Formal Methods to Web Services", with reference TIN2006-15578-C02-02, and the JCCM regional project "Application of formal methods to the design and analysis of Web Services and e-commerce" (PAC06-0008-6995)

---

*Please use the following format when citing this chapter:*

Tobarra, L., Cazorla D., Cuartero, F., Díaz, G., Cambronero, E., 2007, in IFIP International Federation for Information Processing, Volume 248, Wireless Sensor and Actor Networks, eds. L. Orozco-Barbosa, Olivares, T., Casado, R., Bermudez, A., (Boston: Springer), pp. 95-106.

In order to design a secure network, several aspects have to be considered [1]: Key establishment and trust setup, secrecy and authentication, and privacy. Key establishment can be considered the base of the system; a secure and efficient key distribution mechanism is needed for large scale sensor networks. Once every node has its own keys, these are used to authenticate and encrypt (if needed) the messages they exchange. Several protocols have been proposed in the literature related to authentication and privacy [2,3], and key distribution [4,5,6]

In this paper we have focused in two of these protocols: TinySec [2] in the field of authentication and encryption, and LEAP [4] in the field of key management. TinySec is a fully-implemented link layer security architecture for wireless sensor networks included in the official TinyOS [7] release. LEAP is a powerful keying mechanism that supports the establishment of four types of keys for each sensor node with little intervention of the base station. A brief overview of these protocols is given in Section 2.

In line with the development of security protocols, some techniques have also been developed to model a system and check properties on it. One of the most promising techniques in this line is *model checking*. Model checking [8] is a formal methods based technique for verifying finite-state-concurrent systems, and has been implemented in several tools. One of the main advantages of this technique is that it is automatic and allows us to see if a system works as expected. In case the system does not work properly, the model checking tool provides a trace that leads to the source of the error.

Model checking has become a key point in the design of concurrent and distributed system because it allows us to ensure the correctness of a design at the earliest stage possible. Model checking has two main advantages over two classical techniques such as simulation and testing: *i*) we do not need to build a prototype of the system, and *ii*) we are able to verify the system against every single execution trace. The latter is very important because using simulation or testing we can only find errors, but we cannot ensure that the whole system behaves as expected (some errors may remain hidden until the system is in production stage).

Some general purpose model checking tools have been developed by different research groups: Spin, UPPAAL, Mur $\phi$ . These tools allow us to verify not only the functional properties of a system (e.g. Spin), but also the performance of a real-time system (e.g. UPPAAL). Although we can use these general purpose tools in order to verify security protocols, we consider that it is preferable (and more intuitive) to use a tool devoted to the verification of security protocols. Among these tools we can find Casper/FDR2 toolbox [9] and AVISPA [10].

The use of model checking tools to verify security protocols has been successful in the past in different areas such as Web Services [11,12,13], Instant Messaging [14], or Transport Layer security protocols [15,16].

In this paper, we present a formal verification of wireless sensor security protocols using AVISPA (Automated Validation of Internet Security Protocol and Applications) framework. AVISPA provides a high-level formal language HLPSL [17] for specifying protocols and their security properties. Once we have

specified the model of the system, AVISPA translates it into an intermediate format IF. This is the input of several backends that are integrated into AVISPA framework: SATMC [18] (the one used in this analysis), OFMC, Cl-Atse and TA4SP. Besides, only one model is specified although it can be analysed with the four backends. AVISPA also offers a graphical interface SPAN [19] that helps in the specifying task.

Providing adequate security in wireless networks is not an easy task due to its broadcast nature. An intruder can overhear, intercept messages, inject new messages or modify messages in transit. This kind of intruder is called Dolev-Yao Intruder [20]. The intruder implemented in AVISPA is a Dolev-Yao intruder, which is appropriate to analyse wireless security protocols.

The paper is organised as follows. In Section 2 a brief overview of TinySec and LEAP is given. Section 3 is devoted to the formal verification of TinySec and LEAP, where four scenarios have been considered depending on the key distribution mechanism used. Finally in Section 4 we give our conclusions and future work.

## 2 TinySec + LEAP

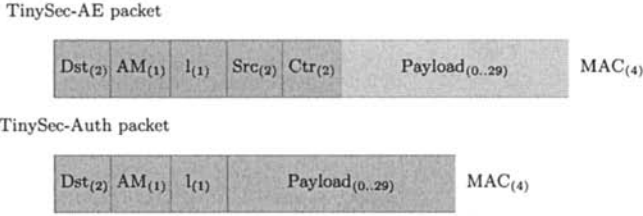
TinySec [2] is a fully-implemented link layer security architecture for wireless sensor networks. The design of TinySec was based on existing security primitives proven to be secure. Using these primitives, a lightweight design was made taking into account wireless sensor networks particularities, mainly limited computation and communication capabilities, as well as low power consumption. TinySec is part of the official TinyOS [7] release.

The main goals of TinySec are performance, usability and security. Inside the security aspects, three main goals are considered: access control, message integrity, and confidentiality. Outside the scope of TinySec is to avoid replay attacks, which is left to higher layers of the protocol stack.

TinySec considers two operations with application layer data: authentication and semantically secure encryption. TinySec authenticates a packet using a message authentication code, CBC-MAC [21]. Semantically secure encryption<sup>1</sup> is made using an 8 byte initialisation vector (IV) and cipher block chaining (CBC) as encryption scheme [22]. Taking into account both operations, two kinds of packets can be found: TinySec-AE, that offers authentication and encryption, and TinySec-Auth, that offers only authentication. A detailed view of both kinds of packets is shown in Fig. 1 (see [2]).

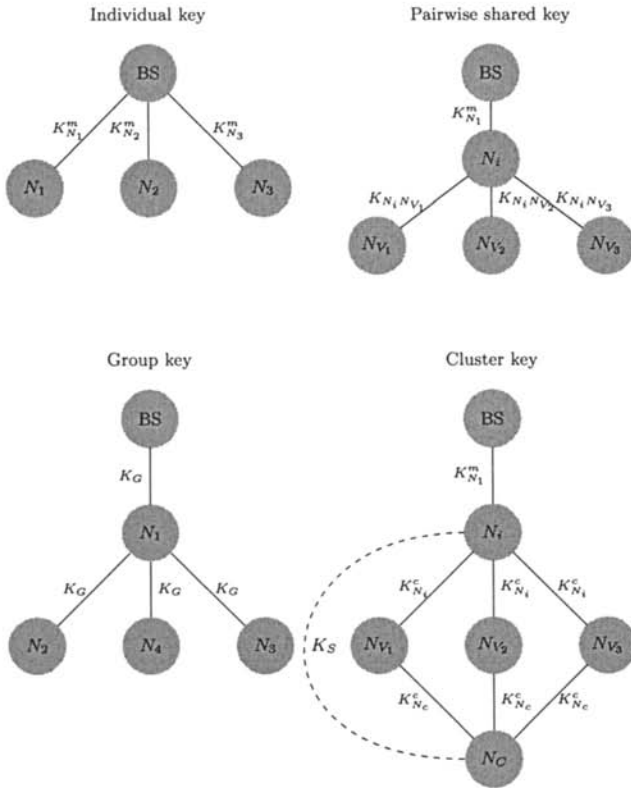
In order to encrypt and decrypt data, shared keys are needed. TinySec does not address the problem of obtaining those keys; any particular keying mechanism can be used in conjunction with TinySec. In this paper we consider Localized Encryption and Authentication Protocol (LEAP) [4] as the keying mechanism. LEAP supports the establishment of four types of keys for each sensor node: an individual key shared with the base station, a pairwise key shared with

<sup>1</sup> Encrypting the same plaintext two times should give two different cipher-texts



**Fig. 1.** TinySec packet formats: TinySec-AE and TinySec-Auth

another sensor node, a cluster key shared with multiple neighboring nodes, and a group key that is shared by all the nodes in the network. One interesting feature of LEAP is that it minimises the involvement of the base station. Fig. 2 shows the four LEAP keying mechanisms.



**Fig. 2.** LEAP keying mechanisms

### 3 Verification of TinySec + LEAP

The combination of TinySec and LEAP allows us to build a complete solution, where LEAP is responsible for obtaining the most convenient shared key at every moment, and TinySec is responsible for the authentication and encryption of messages exchanged between nodes. Thus, we have considered four different configurations of TinySec and LEAP, depending on the key mechanism used, and the kind and the situation of the nodes that communicate with each other:

- Case A. Request from the base node to a normal node using an individual key using messages TinySec-AE.
- Case B. Request from the base node to a normal node using an individual key using messages TinySec-Auth.
- Case C. Communication between immediate neighbouring nodes using a pairwise shared key.
- Case D. Communication between immediate neighbouring nodes using a cluster key.
- Case E. Communication between non immediate neighbouring nodes nodes using a cluster key.

During all these analysis we will consider that before deployment, a node master key  $K_m$  has been saved inside every node in the network. We also adopt the Dolev-Yao intruder model, where an intruder can overhear, intercept, alter, or inject any messages into the radio communication channel.

*Case A.* The network configuration is shown in Fig. 3. In this case a base node ( $BS$ ) makes a request to a normal node ( $N_i$ ) which has an individual (unique) key that it shares with the base station

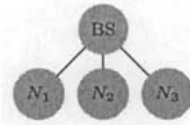


Fig. 3. Sensor network: cases A and B.

The protocol for TinySec-AE packets using Avispa syntax is as follows:

1.  $BS \rightarrow N_i: IV_1\{(IV_1 \oplus Data_1)\}_{F(K_m.N_i)} \cdot \{MAC(IV_1.Data_1)\}_{H(K_m.N_i)}$   
 where  $IV_1 = N_i.AM_1.Size_1.BS.Counter$
2.  $N_i \rightarrow BS: IV_2.\{(IV_2 \oplus Data_2)\}_{F(K_m.N_i)} \cdot \{MAC(IV_2.Data_2)\}_{H(K_m.N_i)}$   
 where  $IV_2 = BS.AM_2.Size_2.N_i.Suc(Counter)$

In this protocol, functions  $F$  and  $H$  are pseudo-random functions that allow to calculate the encryption key and the authentication key taking as parameters the master key ( $K_m$ ) and the id of the node ( $N_i$ ). Due to the fact that Avispa

does not offer arithmetic semantics, the increase of the counter is represented by a function  $Suc$ , such as  $Suc(0)$  represents 1,  $Suc(Suc(0))$  represents 2 and so on.

The properties we have to analyse are the following:

- Authentication of  $Data_1$  and  $Data_2$ . i.e., the node  $N_i$  and the base station (BS) share the same value for  $Data_1$  and  $Data_2$  and both execute the same session of the protocol. This property allows us to proof that bilateral authentication is achieved by using the MAC, and the integrity of the message is guaranteed.
- Confidentiality of  $Data_1$  and  $Data_2$ , i.e.,  $Data_1$  and  $Data_2$  are secret values shared between  $N_i$  and BS, and they are not known by an intruder or third parties.

The verification with Avispa finds only the following replay attack, where  $I_{BS}$  represents an intruder playing the role of the base station:

1. BS  $\rightarrow$   $N_1$  :  $IV_1.\{(IV_1 \oplus Data_1)\}_{F(K_m.N_1)}. \{MAC(IV_1.Data_1)\}_{H(K_m.N_1)}$   
where  $IV_1 = N_1.AM_1.Size_1.BS.0$
2.  $N_1 \rightarrow$  BS :  $IV_2.\{(IV_2 \oplus Data_2)\}_{F(K_m.N_1)}. \{MAC(IV_2.Data_2)\}_{H(K_m.N_1)}$   
where  $IV_2 = BS.AM_2.Size_2.N_1.Suc(0)$
1.  $I_{BS} \rightarrow$   $N_1$  :  $IV_1.\{(IV_1 \oplus Data_1)\}_{F(K_m.N_1)}. \{MAC(IV_1.Data_1)\}_{H(K_m.N_1)}$   
where  $IV_1 = N_1.AM_1.Size_1.BS.0$
2.  $N_1 \rightarrow$  BS :  $IV_3.\{(IV_3 \oplus Data_3)\}_{F(K_m.N_1)}. \{MAC(IV_3.Data_3)\}_{H(K_m.N_1)}$   
where  $IV_3 = BS.AM_3.Size_3.N_1.Suc(0)$

Nevertheless, as was said before, TinySec does not manage replay attacks, which are left to higher layers of the protocol stack. Apart from this attack, the protocol is secure, even when a node is compromised by the intruder.

*Case B.* In this case we use the same scenario than in the previous case (see Fig. 3), but we consider TinySec-Auth messages instead of TinySec-AE messages. The protocol for TinySec-AE packets using Avispa syntax is as follows:

1. BS  $\rightarrow$   $N_i$ :  $N_i.AM_1.Size_1.Data_1. \{MAC(N_i.AM_1.Size_1.Data_1)\}_{H(K_m.N_i)}$
2.  $N_i \rightarrow$  BS:  $BS.AM_2.Size_2.Data_2. \{MAC(BS.AM_2.Size_2.Data_2)\}_{H(K_m.N_i)}$

As we mentioned before, TinySec-Auth does not provide any confidentiality mechanisms. Thus, we can only analyse the authentication of  $Data_1$  and  $Data_2$ , i.e., we can proof the bilateral authentication between BS and  $N_i$ , by means of the MAC messages, and also the integrity of messages. As in the previous case, we have found a replay attack that we omit

*Case C.* The network configuration is shown in Fig. 4. In this case a node ( $N_1$ ) shares a pairwise key with each of its immediate neighbours ( $N_2$  and  $N_3$ ). The protocol for TinySec-AE packets using Avispa syntax is as follows, where  $A$  represents node  $N_1$  and  $B$  represents one of its neighbours ( $N_2$  or  $N_3$ ):

1.  $A \rightarrow B : A, Nonce_A$
2.  $B \rightarrow A : B.\{MAC(Nonce_A.B)\}_{H(K_m.B)}$
3.  $A \rightarrow B : B.AM_1.Size_1.Data_A.\{MAC(B.AM_1.Size_1.Data_A)\}_{K'_{AB}}$   
 where  $K'_{AB} = H(H(K_m.B).A)$
4.  $B \rightarrow A : IV_2.\{(IV_2 \oplus Data_B)\}_{K_{AB}}.\{MAC(IV_2.Data_B)\}_{K'_{AB}}$   
 where  $IV_2 = A.AM_2.Size_2.B.Counter$   
 and  $K_{AB} = F(F(K_m.N_2).N_1)$

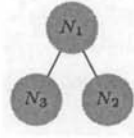


Fig. 4. Sensor network: cases C and D

The properties we have to analyse are the following:

- Authentication of  $Nonce_A$ ,  $Data_A$  and  $Data_B$ . i.e., nodes  $A$  and  $B$  share the same value for  $Nonce_A$ ,  $Data_A$  and  $Data_B$  and both execute the same session of the protocol.
- Confidentiality of  $Data_B$ , i.e.,  $Data_B$  is a secret value shared between  $A$  and  $B$ , and remains unknown to an intruder or third parties.

In this case Avispa finds a *man-in-the-middle* attack, where  $I_A$  represents the intruder playing the role of node  $A$ ,  $I_{B_2}$  represents the intruder playing the role of node  $B$ ,  $B_1$  represents node  $B$  communicating with the intruder, and  $B_2$  represents node  $B$  communicating with  $A$ :

1.  $I_A \rightarrow B_1 : A, Nonce_I$
2.  $B_1 \rightarrow I_A : B.\{MAC(Nonce_I.B)\}_{H(K_m.B)}$
1.  $A \rightarrow I_{B_2} : A, Nonce_A$
2.  $I_A \rightarrow B_2 : N_x.Nonce_A$
2.  $B_2 \rightarrow A : B.\{MAC(Nonce_A.B)\}_{H(K_m.B)}$
3.  $A \rightarrow I_{B_2} : B.AM_1.Size_1.Data_A.\{MAC(B.AM_1.Size_1.Data_A)\}_{K'_{AB}}$
3.  $I_A \rightarrow B_1 : B.AM_1.Size_1.Data_A.\{MAC(B.AM_1.Size_1.Data_A)\}_{K'_{AB}}$
4.  $B_1 \rightarrow I_A : A.AM_2.Size_2.B.Counter.\{(IV_2 \oplus Data_B)\}_{K_{AB}}.\{MAC(IV_2.Data_B)\}_{K'_{AB}}$

First the intruder, playing the role of  $A$ , starts the protocol with  $B$  (denoted  $B_1$ ), and sends a false nonce, which is answered by  $B$  ( $B_1$ ). Then,  $A$  starts a session with  $B$  ( $B_2$ ) but this message is intercepted by the intruder ( $I_{B_2}$ ) which modifies the message and sends to  $B$  ( $B_2$ ) the identity of a false node  $N_x$  and the nonce of  $A$ . Node  $B$  ( $B_2$ ) sends the answer of the last message to  $A$ , and  $A$  responds with a request of data ( $Data_2$ ) to  $B$  ( $B_2$ ). Again, the request is intercepted by the intruder who redirects the message to  $B$  ( $B_1$ ). At this moment,  $B$  ( $B_1$ ) thinks that it has received a correct request from  $A$ , and then it sends  $Data_B$  to the intruder playing the role of  $A$  ( $I_A$ )

At the end we conclude that  $B_1$  has exchanged information ( $Data_2$ ) with the intruder, and  $B_2$  thinks that it has talked to a node  $N_x$  that does not exist. A solution to this attack consists in authenticate not only the answer from  $B$  in message 2 but also the message 1 sent from  $A$ . The modified version of the protocol is:

1.  $A \rightarrow B : A.Nonce_A.MAC(A.Nonce_A)_{H(K_{m,A})}$
2.  $B \rightarrow A : B.\{MAC(Nonce_A.B)\}_{H(K_{m,B})}$
3.  $A \rightarrow B : B.AM_1.Size_1.Data_A.\{MAC(B.AM_1.Size_1.Data_A)\}_{K'_{AB}}$   
where  $K'_{AB} = H(H(K_{m,B}).A)$
4.  $B \rightarrow A : IV_2.\{(IV_2 \oplus Data_B)\}_{K_{AB}}.\{MAC(IV_2.Data_B)\}_{K'_{AB}}$   
where  $IV_2 = A.AM_2.Size_2.B.Counter$  y  $K_{AB} = F(F(K_{m,N_2}).N_1)$

*Case D.* Sensor network configuration is shown in Fig. 4. In this case a node ( $N_1$ ) shares a cluster key with each of its immediate neighbours ( $N_2$  and  $N_3$ ).

The protocol for TinySec-AE packets using Avispa syntax is as follows, where  $A$  represents node  $N_1$ ,  $B$  represents one of its neighbours ( $N_2$  or  $N_3$ ), and  $K_c$  is the cluster key:

1.  $A \rightarrow B : B.AM_1.Size_1.A.Counter.\{(IV_1 \oplus K_c)\}_{K_{AB}}.\{MAC(IV_1.K_c)\}_{K'_{AB}}$   
where  $IV_1 = B.AM_1.Size_1.A.Counter$ ,  $K_{AB} = F(F(K_{m,B}).A)$  and  $K'_{AB} = H(H(K_{m,B}).A)$
2.  $B \rightarrow A : BS.AM_2.Size_2.done.\{MAC(BS.AM_2.Size_2.done)\}_{H(K_c)}$
3.  $A \rightarrow B : B.AM_1.Size_1.Data_A.\{MAC(B.AM_1.Size_1.Data_A)\}_{H(K_c)}$
4.  $B \rightarrow A : IV_2.\{(IV_2 \oplus Data_B)\}_{F(K_c)}.\{MAC(IV_2.Data_B)\}_{H(K_c)}$   
where  $IV_2 = A.AM_2.Size_2.B.Suc(Counter)$

The properties we have to analyse are the following:

- Authentication of  $K_c$ ,  $Data_A$  and  $Data_B$ . i.e., nodes  $A$  and  $B$  share the same value for  $K_c$ ,  $Data_A$  and  $Data_B$  and both execute the same session of the protocol.
- Confidentiality of  $Data_B$  and  $K_c$ , i.e.,  $Data_B$  and  $K_c$  are secret values shared between  $A$  and  $B$ , and they remain unknown to an intruder or third parties.

After analysing the protocol with Avispa, an interesting attack based on types was found:

1.  $A \rightarrow B : B.AM_1.Size_1.A.Counter.\{(IV_1 \oplus K_c)\}_{K_{AB}}.\{MAC(IV_1.K_c)\}_{K'_{AB}}$   
where  $IV_1 = B.AM_1.Size_1.A.Counter$ ,  
 $K_{AB} = F(F(K_{m,B}).A)$  and  $K'_{AB} = H(H(K_{m,B}).A)$
2.  $B \rightarrow I_A : BS.AM_2.Size_2.done.\{MAC(BS.AM_2.Size_2.done)\}_{H(K_c)}$
3.  $I_A \rightarrow B : BS.AM_2.Size_2.done.\{MAC(BS.AM_2.Size_2.done)\}_{H(K_c)}$
4.  $B \rightarrow I_A : IV_2.\{(IV_2 \oplus Data_B)\}_{F(K_c)}.\{MAC(IV_2.Data_B)\}_{H(K_c)}$   
where  $IV_2 = A.AM_2.Size_2.B.Suc(Counter)$

In this attack the intruder intercepts the message sent from  $B$  to  $A$  in step 2. In step 3, the intruder sends the intercepted message back to  $B$  as if was a true request from  $A$  to  $B$ .  $B$  takes the message as a request and misunderstands the label *done* as it was  $Data_A$ . In step 4,  $B$  sends  $Data_B$  to the intruder.



This is a type flaw attack, i.e., type checking has not been done and a constant label has been interpreted as a variable data. One solution to this attack has been proposed by Heather et al. in [23] which basically consists in tagging each field with information about its type, although this solution could not be adequate in wireless sensor networks because it adds some extra bits of information into each message. In real implementations of the protocol, programmers should take into account this type flaw attack and do type checking in order to avoid a possible attack. In any case, this kind of attack can be a problem because network bandwidth could be saturated and there is a consumption of resources in the node.

*Case E.* Sensor network configuration is shown in Fig. 5. In this case a node (e.g.  $N_1$ ) shares a cluster key with non immediate neighbouring nodes (e.g.  $N_4$  and  $N_5$ ).

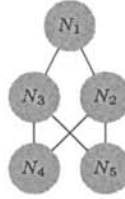


Fig. 5. Sensor Network: Case E

The protocol for TinySec-AE packets using Avispa syntax is as follows, where  $A$  represents node  $N_1$ ,  $B$  represents a non neighbour node ( $N_4$  or  $N_5$ ),  $N_i$  represents a neighbour node ( $N_2$  and  $N_3$ ):

- Q.  $A \rightarrow N_i : A, B$   
 R.  $N_i \rightarrow A : A.AM_1.Size_1.N_i.\{MAC(A.AM_1.Size_1.N_i)\}_{K'_{AN_i}}$   
 1.  $A \rightarrow N_i : IV_2.F(SK_i, 0).\{(IV_2 \oplus SK_i)\}_{K_{AN_i}}.\{MAC(IV_2.SK_i.F(SK_i, 0))\}_{K'_{AN_i}}$   
     where  $IV_2 = N_i.AM_2.Size_2.A.Counter$   
 2.  $N_i \rightarrow B : IV_3.F(SK_i, 0).\{(IV_3 \oplus SK_i)\}_{K_{N_iB}}.\{MAC(IV_3.SK_i.F(SK_i, 0))\}_{K'_{N_iB}}$   
     where  $IV_3 = B.AM_3.Size_3.N_i.Counter$   
     When  $B$  has every  $SK_i$   
 D.  $B \rightarrow A : A.AM_4.Size_4.done\{MAC(A.AM_4.Size_4.done)\}_{Sk}$   
     where  $Sk = Sk_1 \oplus \dots \oplus Sk_n$

Avispa only finds a replay attack, that we do not consider. Apart from this attack, the protocol is secure.

## 4 Conclusions and Future Work

In this paper we have presented a formal approach to the security analysis of wireless sensor networks by means of a model checking tool called Avispa. Sev-

eral models of the network have been considered depending on the relative position and roles of the nodes. Two wireless sensor security protocols have been considered in order to build a complete solution: TinySec, who is in charge of the authentication and encryption of messages, and LEAP, who covers the key distribution mechanism.

The five models we have presented have been analysed with Avispa, and we have obtained the following results:

*Case A. Request from the base node to a normal node using an individual key using messages TinySec-AE.* The verification with Avispa finds only a replay attack where an intruder may play the role of the base station. Nevertheless, TinySec does not manage replay attacks, which are left to higher layers of the protocol stack. Apart from this attack, the protocol is secure,

*Case B. Request from the base node to a normal node using an individual key using messages TinySec-Auth.* As in the previous case, Avispa finds only a replay attack. Under our previous assumptions about replay attacks, we can consider that the protocol is secure,

*Case C. Communication between immediate neighbour nodes using a pairwise shared key.* In this case Avispa finds a *man-in-the-middle* attack where the intruder may play at the same time the role of two nodes in order to obtain real information from one of them. Consequently, confidentiality is lost. A solution to this attack is proposed and consists in authenticate the first message sent from the initiator (node A).

*Case D. Communication between immediate neighbour nodes using a cluster key.* In this case a type flaw attack is found. As in the previous case, the intruder can obtain real data from one of the nodes, and therefore confidentiality is lost. A solution to this kind of attack has been proposed in [23]; nevertheless this solution is not adequate in wireless sensor networks because increases the overload of security protocols in messages.

*Case E. Communication between non immediate neighbour nodes using a cluster key.* In this case, apart from a replay attack, the protocol is secure.

Our future work is concerned with extending our analysis to other security protocols for wireless sensor networks such as SNEP,  $\mu$ TESLA (both defined in [3]), and MiniSec [24]. Also we are interested in the analysis of TinySec with other distribution key protocols such as LEAP+[25] and TinyPK [26].

## References

1. Perrig, A., Stankovic, J.A., Wagner, D.: Security in wireless sensor networks. *Commun. ACM* **47** (2004) 53–57

2. Karlof, C., Sastry, N., Wagner, D.: TinySec: a link layer security architecture for wireless sensor networks. In: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys 2004, Baltimore, MD, USA, November 3-5, 2004, ACM (2004) 162–175
3. Perrig, A., Szewczyk, R., Tygar, J.D., Wen, V., Culler, D.E.: SPINS: Security protocols for sensor networks. *Wireless Networks* 8 (2002) 521–534
4. Zhu, S., Setia, S., Jajodia, S.: LEAP: efficient security mechanisms for large-scale distributed sensor networks. In Jajodia, S., Atluri, V., Jaeger, T., eds.: ACM Conference on Computer and Communications Security, ACM (2003) 62–72
5. Chan, H., Perrig, A., Song, D.X.: Random key predistribution schemes for sensor networks. In: IEEE Symposium on Security and Privacy, IEEE Computer Society (2003) 197
6. Eschenauer, L., Gligor, V.: A key-management scheme for distributed sensor networks. In Atluri, V., ed.: ACM Conference on Computer and Communications Security, ACM (2002) 41–47
7. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D.E., Pister, K.S.J.: System architecture directions for networked sensors. In: Inter. Conf. on Architectural Support for Programming Languages and Operating Systems, ASPLOS. (2000) 93–104
8. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. The MIT Press (1999)
9. Lowe, G.: Casper: A compiler for the analysis of security protocols. *Journal of Computer Security* 6 (1998) 53–84
10. Armando, A., Basin, D.A., Boichut, Y., Chevalier, Y., Compagna, L., Cuéllar, J., Drielsma, P.H., Héam, P.C., Kouchnarenko, O., Mantovani, J., Mödersheim, S., von Oheimb, D., Rusinowitch, M., Santiago, J., Turuani, M., Viganò, L., Vigneron, L.: The AVISPA tool for the automated validation of internet security protocols and applications. In Etessami, K., Rajamani, S.K., eds.: CAV. Volume 3576 of Lecture Notes in Computer Science., Springer (2005) 281–285
11. Tobarra, M.L., Cazorla, D., Cuartero, F., Diaz, G.: Application of formal methods to the analysis of web services security. In Bravetti, M., Kloul, L., Zavattaro, G., eds.: EPEW/WS-FM. Volume 3670 of Lecture Notes in Computer Science., Springer (2005) 215–229
12. Backes, M., Mödersheim, S., Pfitzmann, B., Viganò, L.: Symbolic and cryptographic analysis of the secure WS-ReliableMessaging scenario. In Aceto, L., Ingólfssdóttir, A., eds.: FoSSaCS. Volume 3921 of Lecture Notes in Computer Science., Springer (2006) 428–445
13. Bhargavan, K., Fournet, C., Gordon, A.D.: Verifying policy-based security for web services. In Atluri, V., Pfitzmann, B., McDaniel, P.D., eds.: ACM Conference on Computer and Communications Security, ACM (2004) 268–277
14. Mannan, M., van Oorschot, P.C.: A protocol for secure public instant messaging. In Crescenzo, G.D., Rubin, A., eds.: Financial Cryptography. Volume 4107 of Lecture Notes in Computer Science., Springer (2006) 20–35
15. Mitchell, J.C.: Finite-state analysis of security protocols. In Hu, A.J., Vardi, M.Y., eds.: CAV. Volume 1427 of Lecture Notes in Computer Science., Springer (1998) 71–76
16. Tobarra, M.L., Cazorla, D., Cuartero, F., Diaz, G.: Formal verification of TLS handshake and extensions for wireless networks. In: Proc. of IADIS International Conference on Applied Computing (AC'06), San Sebastian, Spain, IADIS Press (2006) 57–64

17. Chevalier, Y., Compagna, L., Cuéllar, J., Drielsma, P.H., Mantovani, J., Mödersheim, S., Vigneron, L.: A high level protocol specification language for industrial security-sensitive protocols. In: Proceedings of Workshop on Specification and Automated Processing of Security Requirements (SAPS). (2004) 193–205
18. Armando, A., Compagna, L.: SATMC: A SAT-based model checker for security protocols. In Alferes, J.J., Leite, J.A., eds.: JELIA. Volume 3229 of Lecture Notes in Computer Science., Springer (2004) 730–733
19. Glouche, Y., Genet, T., Heen, O., Courtay, O.: A security protocol animator tool for AVISPA. In: ARTIST2 Workshop on Security Specification and Verification of Embedded Systems, Pisa (2006)
20. Dolev, D., Yao, A.C.C.: On the security of public key protocols. In: FOCS, IEEE (1981) 350–357
21. Bellare, M., Kilian, J., Rogaway, P.: The security of the cipher block chaining message authentication code. *J. Comput. Syst. Sci.* **61** (2000) 362–399
22. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: Proceedings of 38th Annual Symposium on Foundations of Computer Science, IEEE (1997) 394–403
23. Heather, J., Lowe, G., Schneider, S.: How to prevent type flaw attacks on security protocols. *Journal of Computer Security* **11** (2003) 217–244
24. Mark Luk, Ghita Mezzour, A.P., Gligor, V.: Minisec: A secure sensor network communication architecture. In: Proceedings of IEEE International Conference on Information Processing in Sensor Networks (IPSN). (2007)
25. Zhu, S., Setia, S., Jajodia, S.: LEAP : Efficient security mechanisms for large-scale distributed sensor networks. *ACM Transactions on Sensor Networks* **2** (2006) 500–528
26. Watro, R.J., Kong, D., Cuti, S.f., Gardiner, C., Lynn, C., Kruus, P.: TinyPK: securing sensor networks with public key technology. In Setia, S., Swarup, V., eds.: SASN, ACM (2004) 59–64