Telecommunications Systems manuscript No.
(will be inserted by the editor)

Model Checking Wireless Sensor Network Security Protocols: TinySec + LEAP + TinyPK

Llanos Tobarra · Diego Cazorla · Fernando Cuartero · Gregorio Díaz · Emilia Cambronero

Received: date / Accepted: date

Abstract In this paper, a formal analysis of security protocols in the field of wireless sensor networks is presented. Three complementary protocols, TinySec, LEAP and TinyPK, 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

Wireless sensor network have been widely used in many different applications: monitoring building perimeters, battlefield situation data, human vital signs retrieving, reports about malfunctions in a system, etc. These applications highlight the importance of security in wireless sensor networks. Thus, security has become a challenge in wireless sensor due to low capabilities of devices, in terms of computational power and energy consumption, making it difficult to use traditional security protocols.

Two main problems related to security protocols arise. Firstly, the overload that security protocols introduce in messages should be reduced to 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: to reduce the overhead that protocol imposes to messages, and to provide reasonable protection while limiting use of resources.

In order to design a secure network, several aspects have to be considered [17]: 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 [14,18], and key distribution [22,6,10]

In this paper we have focused on three of these protocols: TinySec [14] in the field of authentication and encryption, and LEAP [22] and TinyPK [21] in the field of key management. TinySec is a fully-implemented link layer security architecture for wireless sensor networks included in the official TinyOS [13] release. LEAP is a powerful keying mechanism that supports the establishment of four types of keys for each sensor node with

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)

Instituto de Investigación en Informática
Universidad de Castilla-La Mancha. Albacete, Spain.
E-mail: {mtobarra, dcazorla, fernando, gregorio, emicp}@dsi.uclm.es

little intervention of the base station. One alternative to LEAP is TinyPK [21], that is based on public key technologies adapted to sensor capabilities. 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 its properties. One of the most promising techniques in this line is *model checking*. Model checking [8] is a formal method 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, Murϕ. 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 [15] and AVISPA [1].

The use of model checking tools to verify security protocols has been successful in the past in different areas such as Web Services [19,2,5], or Transport Layer security protocols [16,20].

In this paper, we present a formal verification of wireless sensor security protocols using AVISPA (Automated Validation of Internet Security Protocol an Applications) framework. AVISPA provides a high-level formal language HLPSL [7] 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 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 [11] that helps in the specifying task.

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 [9]. The intruder implemented in AVISPA is a Dolev-Yao intruder, which is appropriate to the analysis of wireless security protocols.

The paper is organised as follows. In Section 2 a brief overview of TinySec, LEAP and TinyPK is given. Section 3 is devoted to the formal verification of TinySec, LEAP and TinyPK, where six 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 and TinyPK

TinySec [14] 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 characteristics, mainly limited computation and communication capabilities, as well as low power consumption. TinySec is part of the official TinyOS [13] 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 [4]. Semantically secure encryption[1] is made using an 8 byte initialisation vector (IV) and cipher block chaining (CBC) as encryption scheme [3]. 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.

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 Localised Encryption and Authentication

[1] Encrypting the same plaintext two times should give two different cipher-texts

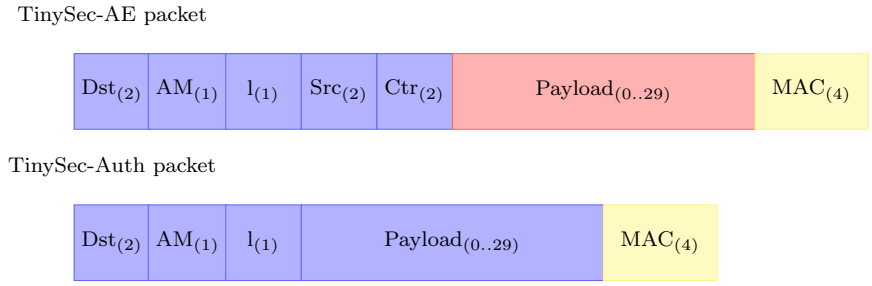

Fig. 1 TinySec packet formats: TinySec-AE and TinySec-Auth

Protocol (LEAP) [22] and TinyPK [21] 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 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.

TinyPK is also focused on supporting confidentiality and authentication for wireless sensor networks. In [21], a protocol is proposed to authenticate a third party (e.g. another sensor network) by means of sensor motes and asymmetric keys. It requires a small public key infrastructure. First, there should be a *Certification Authority*(CA) in the network, such as a base station, which is an entity with a pair of public/private keys trusted by all the network nodes. Each node in the network knows the public CA before deployment. The third party of the protocol should have its own public/private key pair. The protocol operation is divided into two parts: the external party authentication and the node authentication. In the first part, the external party authenticates itself by means of a challenge. After this phase, the node and the external party share the network key and a nonce. In the second phase, the node and the external party establish a new key pair by means of Diffie-Hellman exchange. Each node has a static Diffie-Hellman key, while the third party generates a ephemeral Diffie-Hellman key. A node credential is also exchanged in order to authenticate the node.

3 Verification of TinySec, LEAP and TinyPK

The combination of TinySec and LEAP, or TinySec and TinyPK allows us to build a complete solution, where LEAP or TinyPK are responsible for obtaining the more convenient shared key at every moment, and TinySec is responsible of the authentication and encryption of messages exchanged between nodes. Thus, we have considered six different configurations of TinySec, LEAP and TinyPK, 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.
- Case F. Communication between a network node and third party with asymmetric TinyPK keys.

Throughout this 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

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

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

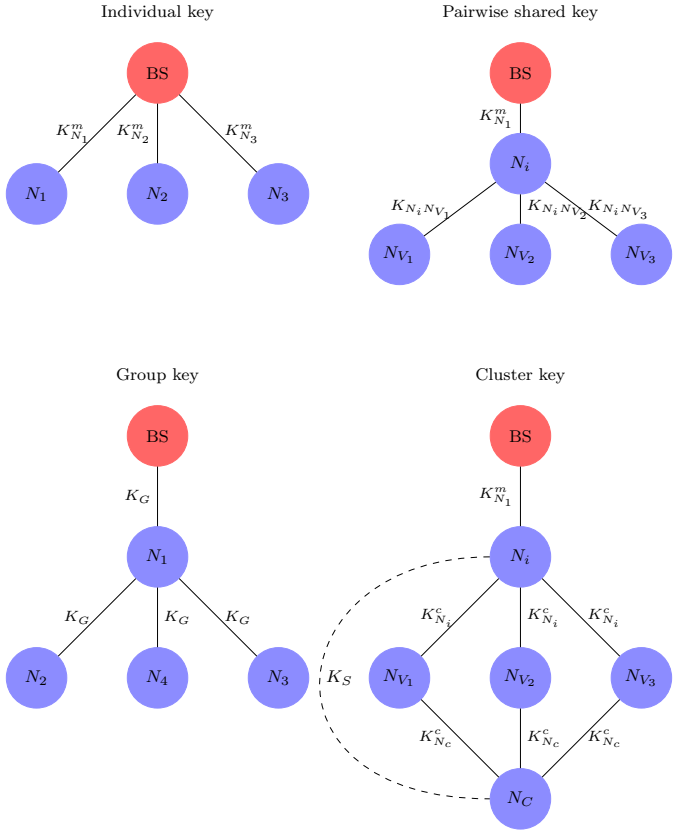

Fig. 2 LEAP keying mechanisms

the master key (K_m) and the id of the node (N_i). Given 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.

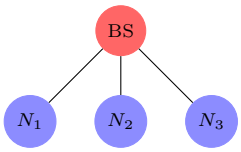

Fig. 3 Sensor network: cases A and B.

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 prove that bilateral authentication is achieved by using the MAC, and the integrity of the message is guarantied.
- 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 replay attack shown in Table 1, where I_{BS} represents an intruder playing the role of the base station. 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 prove 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

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 $IV_3 = BS.AM_3.Size_3.N_1.Suc(0)$

Table 1 Case A. Replay attack

Case C. In this case a node (N_1) shares a pairwise key with each of its immediate neighbours (N_2 and N_3). The network configuration is shown in Fig. 4.

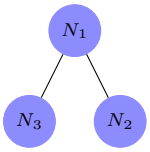

Fig. 4 Sensor network: cases C and D

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 $K'_{AB} = H(H(Km.B).A)$

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 $K_{AB} = F(F(Km.N_2).N_1)$

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 (see Table 2), 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:

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)

Finally we can 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 $K'_{AB} = H(H(Km.B).A)$

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 $K_{AB} = F(F(Km.N_2).N_1)$

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. $I_A \rightarrow B_1 : A, Nonce_I$
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. $B_1 \rightarrow I_A : A.AM_2.Size_2.B.Counter.\ \{(IV_2 \oplus Data_B)\}_{K_{AB}}.\ \{MAC(IV_2.Data_B)\}_{K'_{AB}}$

Table 2 Case C. Man-in-the-middle attack

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 $IV_1 = B.AM_1.Size_1.A.Counter$,
$K_{AB} = F(F(Km.B).A)$ and
$K'_{AB} = H(H(Km.B).A)$

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 (see Table 3). 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 were a true request from A to B. B takes the message as a request an 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 [12] which basically consists of 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).

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 $Sk = Sk_1 \oplus \ldots \oplus Sk_n$

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

Case F. In this case, a third party *EP* establishes an authenticated communication channel with a sensor network. The first three messages are intended to authenticate the third party to the node, while the three last messages are for authenticating the sensor mote with the third party.

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 $IV_1 = B.AM_1.Size_1.A.Counter$
   $K_{AB} = F(F(Km.B).A)$ and
   $K'_{AB} = H(H(Km.B).A)$
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 $IV_2 = A.AM_2.Size_2.B.Suc(Counter)$

Table 3 Case D. Type flaw attack

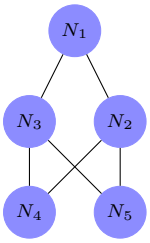

Fig. 5 Sensor Network: Case E

The protocol for TinySec-AE and TinySec-Auth packets in combination with TinyPK using AVISPA syntax is as follows:

1A. $EP \rightarrow N_i : IV_1.\ \{xor(IV_1, PK_{ep})\}_{PK_{ca}}$
where $IV_1 = Size_1.EP.0.Nonce$
2A. $EP \rightarrow N_i : \{MAC(IV_1.Nonce.\ PK_{ep})\}_{inv(PK_{ep})}$
3A. $N_i \rightarrow EP : IV_2.Nonce.\{xor(IV_2, K_m)\}_{PK_{ep}}.$
$\{MAC(IV_2.Nonce.K_m)\}_{PK_{ep}}$
where $IV_2 = EP.Size_2.N.Suc(0)$
1B. $EP \rightarrow N_i : IV_3.Nonce.$
$\{xor(IV_1, DHdym)\}_{F(K_m,EP)}$
where $IV_3 = Size_3.EP.Suc(Suc(0)).Nonce$
2B. $EP \rightarrow N_i : \{MAC(IV_3.Nonce.DHdym)\}_{H(K_m,EP)}$
3B. $N_i \rightarrow EP : IV_4.Credential.$
$\{MAC(Nonce)\}_{DH(DHstatic,DHdym)}.$
$\{MAC(IV_4.Credential)\}_{H(K_m,N)}$
where $IV_4 = EP.Size_4.N.Suc(Suc(Suc(0)))$
and $Credendtial = \{N\}_{inv(PK_{ca})}$

Initially, we assume that the third party EP knows the corresponding public key PK_{ca} of the network Certification Authority (CA), and that it also has its own public key PK_{ep}. The TinyPK proposal assumes that these key pairs are RSA keys.

The messages 1A and 2A are the same challenge message but, due to the size of the keys, the challenge message needs to be split in two TinySec messages. The challenge messages include a challenge nonce and the public key of the third party. The sensor node checks the message authentication code as soon as it receives the first two messages. If it trusts EP, it answers back the challenge message including the network key K_m. After EP receives message 3A, the third party is authenticated. Then the sensor node is authenticated.

The third party EP sends another challenge message, including the previous phase nonce. It also generates ephemeral Diffie-Hellman key $DHdym$ that is added to the challenge message. Each node has a static Diffie-Hellman key $DHstatic$ that is pre-loaded in the node before its deployment. Thus, after the two challenge messages, if the sensor node checks the message authentication code successfully, it generates a new Diffie-Hellman key $DH(DHdym, DHstatic)$. In the challenge response, it includes its credentials and the nonce signed with the new key.

AVISPA only finds a man-in-the-middle attack. In this attack, a sensor node and the third party execute a run of the protocol, but this run is not finished because the intruder deletes the last message. Then, the intruder replays the third party messages to the same sensor node. As the challenge node is computed by means of a timing function, the sensor node should be able to detect that messages are replayed. Thus, we do not consider this attack feasible. 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. Several models of the network have been considered depending on the relative position and roles of the nodes. Three wireless sensor security protocols have been considered in order to build a complete solution: TinySec, which is in charge of the authentication and encryption of messages; LEAP, which covers the key distribution mechanism and TinyPK, which allows to establish an authenticated conversation with an external third party.

The six 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 [12]; nevertheless this solution is not adequate in wireless sensor networks because it 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.

Case F.Communication between a network node and third party with asymmetric TinyPK keys . This case is also secure.

Our future work is concerned with extending our analysis to other security protocols for wireless sensor networks intended to secure routing with time constraints like μTESLA [18]. We are also interested in building a more accurate intruder model taking into account wireless sensor network capabilities and constraints.

References

1. 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: K. Etessami, S.K. Rajamani (eds.) CAV, *Lecture Notes in Computer Science*, vol. 3576, pp. 281–285. Springer (2005)
2. Backes, M., Mödersheim, S., Pfitzmann, B., Viganò, L.: Symbolic and cryptographic analysis of the secure WS-ReliableMessaging scenario. In: L. Aceto, A. Ingólfsdóttir (eds.) FoSSaCS, *Lecture Notes in Computer Science*, vol. 3921, pp. 428–445. Springer (2006)
3. 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, pp. 394–403. IEEE (1997)
4. Bellare, M., Kilian, J., Rogaway, P.: The security of the cipher block chaining message authentication code. J. Comput. Syst. Sci. **61**(3), 362–399 (2000)
5. Bhargavan, K., Fournet, C., Gordon, A.D.: Verifying policy-based security for web services. In: V. Atluri, B. Pfitzmann, P.D. McDaniel (eds.) ACM Conference on Computer and Communications Security, pp. 268–277. ACM (2004)
6. Chan, H., Perrig, A., Song, D.X.: Random key predistribution schemes for sensor networks. In: IEEE Symposium on Security and Privacy, p. 197. IEEE Computer Society (2003)
7. 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), pp. 193–205 (2004)
8. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. The MIT Press (1999)
9. Dolev, D., Yao, A.C.C.: On the security of public key protocols. In: FOCS, pp. 350–357. IEEE (1981)
10. Eschenauer, L., Gligor, V.: A key-management scheme for distributed sensor networks. In: V. Atluri (ed.) ACM Conference on Computer and Communications Security, pp. 41–47. ACM (2002)
11. 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)
12. Heather, J., Lowe, G., Schneider, S.: How to prevent type flaw attacks on security protocols. Journal of Computer Security **11**(2), 217–244 (2003)
13. 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, pp. 93–104 (2000)
14. 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, pp. 162–175. ACM (2004)
15. Lowe, G.: Casper: A compiler for the analysis of security protocols. Journal of Computer Security **6**(1-2), 53–84 (1998)
16. Mitchell, J.C.: Finite-state analysis of security protocols. In: A.J. Hu, M.Y. Vardi (eds.) CAV, *Lecture Notes in Computer Science*, vol. 1427, pp. 71–76. Springer (1998)
17. Perrig, A., Stankovic, J.A., Wagner, D.: Security in wireless sensor networks. Commun. ACM **47**(6), 53–57 (2004)

18. Perrig, A., Szewczyk, R., Tygar, J.D., Wen, V., Culler, D.E.: SPINS: Security protocols for sensor networks. Wireless Networks **8**(5), 521–534 (2002)
19. Tobarra, M.L., Cazorla, D., Cuartero, F., Diaz, G.: Application of formal methods to the analysis of web services security. In: M. Bravetti, L. Kloul, G. Zavattaro (eds.) EPEW/WS-FM, *Lecture Notes in Computer Science*, vol. 3670, pp. 215–229. Springer (2005)
20. 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), pp. 57–64. IADIS Press, San Sebastian, Spain (2006)
21. Watro, R., Kong, D., Cuti, S.F., Gardiner, C., Lynn, C., Kruus, P.: Tinypk: securing sensor networks with public key technology. In: SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks, pp. 59–64. ACM Press, New York, NY, USA (2004). DOI 10.1145/1029102.1029113. URL http://portal.acm.org/citation.cfm?id=1029113
22. Zhu, S., Setia, S., Jajodia, S.: LEAP: efficient security mechanisms for large-scale distributed sensor networks. In: S. Jajodia, V. Atluri, T. Jaeger (eds.) ACM Conference on Computer and Communications Security, pp. 62–72. ACM (2003)