

Analysis of security protocol MiniSec for Wireless Sensor Networks*

Llanos Tobarra, Diego Cazorla, Fernando Cuartero, and Gregorio Diaz

Real Time And Concurrent Systems Group (ReTICS)
University of Castilla-La Mancha
Escuela Politécnica Superior de Albacete
Avd. España s/n 02071-Albacete (Spain)
{mtobarra,dcazorla,fernando,gregorio}@dsi.uclm.es

Abstract. In this paper, a formal analysis of a security protocol in the field of wireless sensor networks is presented. MiniSec is a secure network layer protocol for wireless sensor networks. It provides confidentiality, data authentication, replay protection and weak message freshness with low energy consumption. It can work in two modes –MiniSec-U and MiniSec-B – in order to exploit the different energy requirements of unicast and broadcast communications. MiniSec is modelled in two scenarios using the high-level formal language HLPSL, and verified using the model checking tool Avispa. Two main security properties are checked: authenticity and confidentiality of relevant messages components. The first case is the unicast communication between two nodes in order to exchange secret data. The second case is a broadcast message protocol in a sensor network using MiniSec-B for securing messages.

1 Introduction

Security over wireless networks, and specially in sensor networks, has become an important research topic. A basic secure network layer for sensor networks should guarantee data secrecy, authentication and protection against replay attacks. On the other hand, security solutions should deal with low capabilities of devices, in terms of computational power and energy consumption. These performance requirements 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

* 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)

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.

In order to design a secure network, several aspects have to be considered [20]: 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 [14,21], and key distribution [26,7,12].

In this paper, we have focus in one of these protocols: MiniSec[18]. MiniSec is a secure network protocol for wireless sensor networks. It uses a only block cipher mode, OCB encryption mode[22], for data secrecy and authentication. To prevent replay attacks, it includes the last bits of a synchronized incremental counter. It works in two communication modes, unicast and broadcast, for exploiting the different energy levels of each type of communication. MiniSec works in two modes, MiniSec-U for securing unicast messages, and MiniSec-B, for securing broadcast messages. MiniSec does not address a key distribution or computing algorithm. But it suggests LEAP [26] as possible key management solution. A brief overview of these protocols is given in Section 3.

We analyse MiniSec using AVISPA (Automated Validation of Internet Security Protocol an Applications) framework [1]. AVISPA is a model checking toolbox that allows an user to analyse security properties of a protocol specification with four different backends. A more detailed description of Avispa is given in Section 2.

The paper is organised as follows. In Section 2 an introduction of formal methods for the analysis of security protocols is presented. In Section 3 a brief overview of MiniSec and LEAP is given. Section 4 is devoted to the formal verification of MiniSec-U mode and MiniSec-B mode. Finally in Section 5 we give our conclusions and future work.

2 Formal Methods for the analysis of security protocols

As new security protocols are designed, new 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 [10] is a technique based on formal methods 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 ϕ , FRD2.0, etc. 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 [17] 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 [23,3,5], wireless security protocols [15], or Transport Layer security protocols [19,24].

2.1 AVISPA toolbox

AVISPA provides a high-level formal language HLP ϕ SL [8] (see fig. 1) 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.

SAT-based Model-Checking (SATMC)[2] translates automatically from security protocol specifications into propositional logic which can be effectively used to find attacks to protocols. In other words, given as input the specification of a security protocol and the associated security property, SATMC generates a sequence of propositional formulae whose models (quickly found by means of state-of-the-art SAT-solvers) correspond to attacks on the protocol.

On-The-Fly model checker (OFMC)[4] is a symbolic model checker that combines two ideas for analyzing security protocols based on lazy, demand-driven search. The first is the use of lazy data types as a simple way of building efficient on-the-fly model checkers for protocols with very large, or even infinite, state spaces. The second is the integration of symbolic techniques and optimizations for modeling a lazy Dolev-Yao intruder whose actions are generated in a demand-driven way.

Constraint Logic (CL-Atse) based Model-Checking of Security Protocols (CL-Atse)[9] is an OCaml-based implementation of the deduction rules. These rules allows anyone to interpret and automatically “execute” a protocol in every possible ways against a generic intruder with the Dolev-Yao deductive capabilities.

TA4SP is an approach based on rewriting on regular tree languages to the security protocols verification inside the European AVISPA project.

Besides, only one model is specified although it can be analysed with the four backends. AVISPA also offers a graphical interface SPAN [13] that helps in the specifying task.

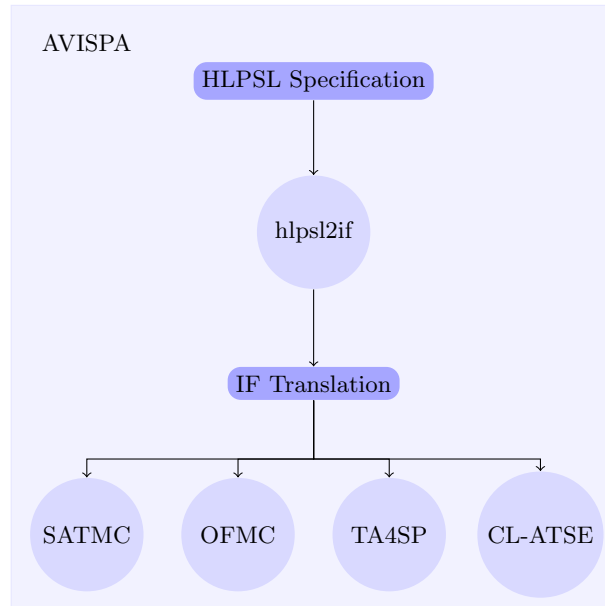


Fig. 1. AVISPA architecture

In AVISPA we find two kind of roles; basic roles and composed roles. A *basic role* describes a protocol participant initial knowledge, its initial state and a set of transitions that describes the behaviour of the participant. Transitions are more expressive notation than A-B notation. They allow to express control flow elements such as if-then-else structures, loops, etc. The basic roles can be composed in order to describe protocol sessions, which are composed roles.

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 [11]. The intruder implemented in AVISPA is a Dolev-Yao intruder, which is appropriate to analyse wireless security protocols.

2.2 Notation

We are going to use the following notation to describe the protocols and the models:

A, B, C	communicating nodes
K_{ab}	Symmetric key shared between nodes A and B
K_{net}	Network key shared among the network nodes
C_{ab}	Shared counter between A and B
N_A	Nonce generated by A
$MAC(M)$	Message Authentication Code function computed over message M
$F(M)$	One-way hash function computed over message M
$Suc(C)$	Succesor of one natural, means $C + 1$
$\{M\}_K$	Message M encrypted with key K
$Chan!M$	Sending message M on channel $Chan$
$Chan?M$	Receiving message M on channel $Chan$

3 MiniSec and LEAP

MiniSec[18] is a secure network layer protocol for wireless sensor networks. The objective of MiniSec is providing confidentiality, data authentication, replay protection and weak freshness.

Confidentiality prevents unauthorized nodes knowing secret data. It is accomplished by means of encryption. MiniSec applies OCB encryption [22] in order to guarantee data confidentiality and data authentication. Let it be $H.M$ a message, it needs that $H.M$ will be authenticated and M will be protected with the symmetric K . The final output of $OCB_K(Nonce, M, H)$ is a ciphertext $\{Nonce, M\}_K$ and a tag τ . The receiver of the message will compute the tag τ in order to check the message integrity.

To fulfill semantic confidentiality¹, instead of initial vector or random data, shared counters are used. MiniSec introduces the *LB Optimization* (Last Bits Optimization) where only the last x bits of the shared counter are sent in a message. A node can determinate when a message is out-of-order or replayed and it can drop it. So, counters enforce a message ordering.

In addition, MiniSec tries to minimize the energy overhead. But, on the other hand, it introduces high node memory requirements.

It works in two modes: unicast mode, called *MiniSec-U* and broadcast mode, called *MiniSec-B*. *MiniSec-U* secure messages between two nodes A and B . Each pair of nodes share a pair of symmetric keys K_{ab} , to encrypt the messages from A to B , and K_{ba} , to encrypt the messages from B to A . They also share an incremental counter assigned to each key.

MiniSec-B also uses OCB encryption, but instead of synchronized counters, it describes two mechanism, a timing-based approach and a bloom-filter approach, that defend against replay attacks. Firstly, the time is divided into t -length

¹ Semantic confidentiality means that if the same plain-text is ciphered twice, the two resulting ciphertexts are different

epochs E_1, E_2, \dots . Using the current epoch or the previous epoch as nonce for OCB encryption, the replay of messages from older epochs is avoided.

The timing-approach is incremented with bloom-filter approach in order to prevent replay attacks inside the current epoch. A bloom-filter[6] is a data structure with two possible operations: membership addition and membership query. It is composed by an array of m bits and a set of hash functions k . Initially, all the array elements are set to zero. Each hash functions associates a new element to an array position which is set to one. If we want to know if an element belongs to the bloom-filter, we should compute its k positions with the hash functions. If any of the k positions is zero, the element is not a member of the bloom filter.

MiniSec-B uses as nonce element in OCB encryption and bloom-filter key the string $nodeID.E_i.Cab$, where $nodeID$ is the sender node identifier, E_i is the current epoch and Cab is a shared counter. Every time that a node receives a message, it checks if it belongs to its bloom filter. If the message is not replayed, it is stored in the bloom filter. Else, the node drops it.

In order to encrypt and decrypt data, shared keys are needed. MiniSec does not address the problem of obtaining those keys; any particular keying mechanism can be used in conjunction with MiniSec. In this paper we consider Localized Encryption and Authentication Protocol (LEAP) [26] 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.

In particular, we are going to consider LEAP pairwise key shared with another sensor node for MiniSec-U mode and LEAP cluster key for MiniSec-B mode.

4 Analysis of MiniSec

We have considered two cases on one network configuration (see fig. 3), depending on the situation of the nodes that communicate with each other:

- *MiniSec-U*: unicast communication between two nodes.
- *MiniSec-B*: broadcast communication.

It is assumed that the network is in a stable phase. The nodes do not dynamically connect to the network or they dynamically disconnect from the network. This assumption simplifies the analysis and it abstracts away the routing problems. We also assume a generic network protocol message format $A.B.H.M$. H is the heading of the message and it contains all control information. M will be the payload of the message. A is the message sender and B is the intended receiver.

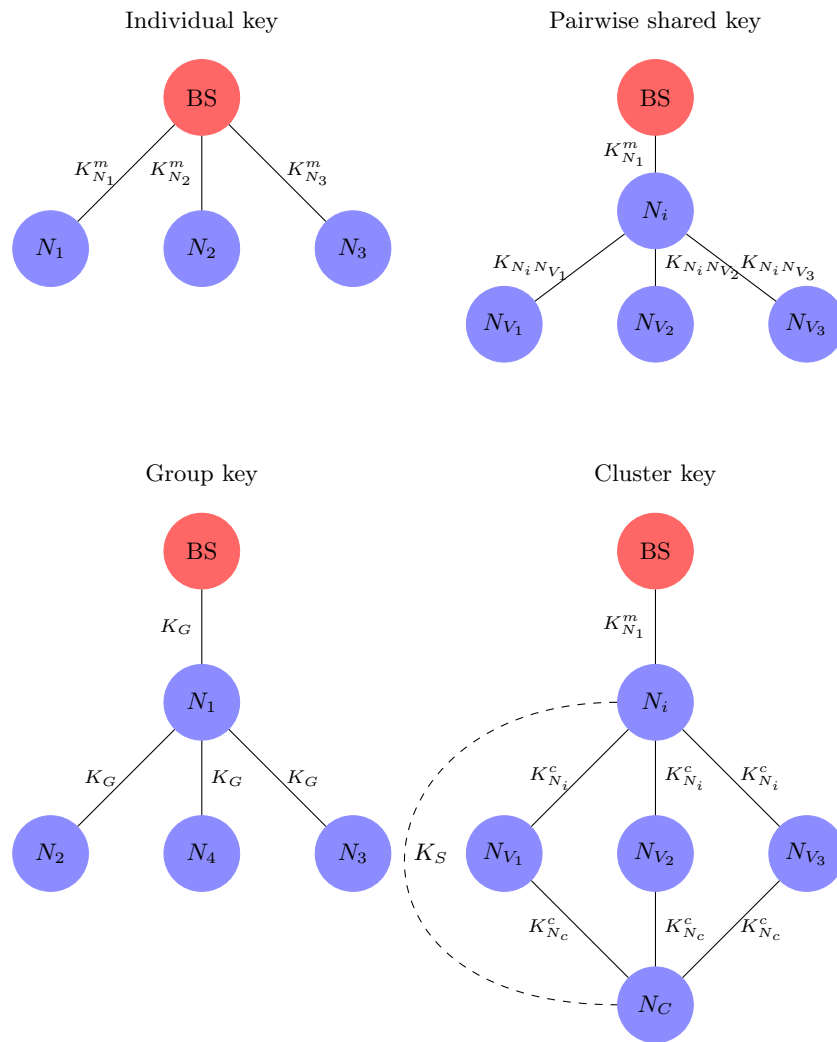


Fig. 2. LEAP keying mechanisms

4.1 Minisec-U

In this case, a node A wants to send a request to its neighbour nodes B and C . The following protocol narration describes the steps between A and a neighbour node B :

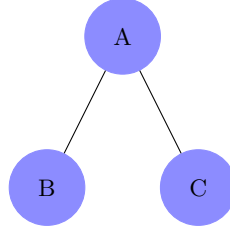


Fig. 3. Sensor network for the two analysed cases

1. $A \rightarrow B : Cab, A, B, H1, Na,$
 $\{MAC(Cab, A, B, H1, Na)\}_{K_a}$
2. $B \rightarrow A : Cba, B, A, H2, Nb,$
 $\{MAC(Cba, B, A, H2, Nb, Na)\}_{K_b}$
3. $A \rightarrow B : Suc(Cab), A, B, H3, \{P3, Suc(Cab)\}_{K_{ab}},$
 $\{MAC(Suc(Cab), A, B, H3,$
 $\{P3.Suc(Cab)\}_{K_{ab}})\}_{K_{ab}}$
4. $B \rightarrow A : Suc(Cba), B, A, H4, \{P4, Suc(Cba)\}_{K_{ba}},$
 $\{MAC(Suc(Cba), B, A, H4,$
 $\{P4, Suc(Cba)\}_{K_{ba}})\}_{K_{ba}}$

where $K_a := F(K_n, A)$, $K_b := F(K_n, B)$, $K_{ab} := F(K_b, A)$ y $K_{ba} := F(K_a, B)$. Each node has a master key K_n . This key has been store in each node memories before node is deployed. A node can play two roles, initiator node A or the responser node B , in this scenario.

The two first steps are devoted to establish a pair of symmetric keys K_{ab} and K_{ba} between the sender node A and its neighbour B . During these two first messages, the nodes do not exchange any secret data. They only guarantee the strong freshness of the request with the nonces Na and Nb .

Then, A sends a request to node B . In this case, the message is authenticated and encrypted with OCB. As we mention in Sec. 3, the output of $OCB_K(H, M, N)$ is a ciphertext $\{M, N\}_K$ and a tag τ . It is evident that we represent the ciphertext as $\{M, N\}_K$, where N are the last x bits of the counter. The tag τ is the message authentication code. This authentication code is represented as $\{MAC(H, \{M, N\}_K)\}_K$, where MAC is an one-way hash function. It is worthy to mention why we do not model the authentication code as only $MAC(H, \{M, N\}_K)$. If a client generates a message $H.M$, the resulting secure message would be $N, H, \{M, N\}_K, MAC(N, H, \{M, N\}_K, .)$. The intruder can modify this secure message in the following way $N, BadH, \{M, N\}_K, MAC(N, BadH, \{M, N\}_K, .)$.

Once the node B receives the request, it answers with some secret data $P4$ to the node A . The fig. 4 represents the diagram transitions for the node A (sender node) and the node B (receiver node).

The properties we have to analyse are the following:

- Authentication of $H3, H3, P4$ and $P4$, i.e., the node A , the node B share the same value for $H3, H3, P4$ and $P4$ and all of them execute the same session

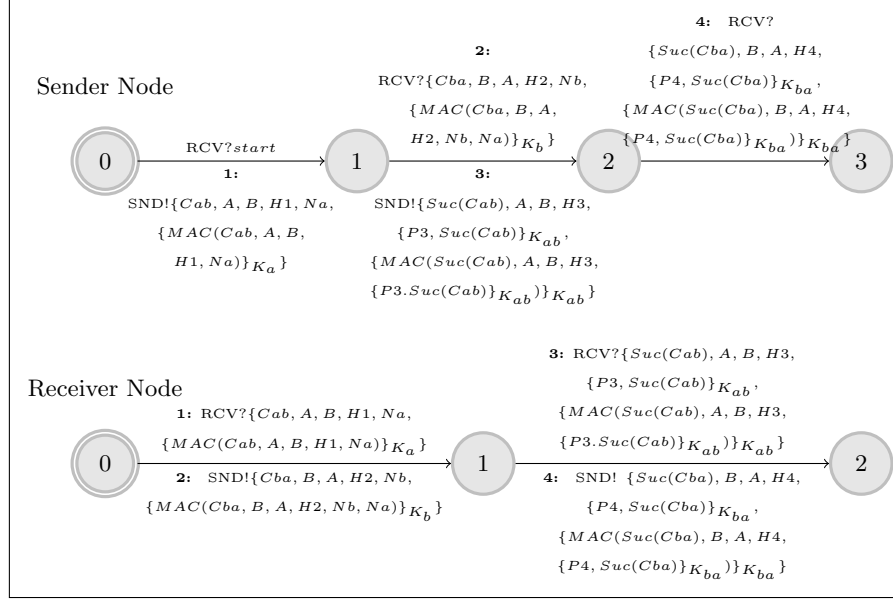


Fig. 4. MiniSec-U model diagram

of the protocol. This property allows us to proof that there are not replay attacks and the freshness of messages is guaranteed.

- Confidentiality of $P3$ and $P4$, i.e., $P3$ and $P4$ are secret values shared between A and B , and they are not known by an intruder or third parties.

We consider two executions of the protocol where there is a sender node A and two receiver nodes B and C . We also analyse a scenario where one of the receiver nodes is the intruder. AVISPA does not report about any attack in any of the cases.

4.2 Minisec-B

In this case, a node A establish a cluster key among the network nodes and it broadcast a message. The protocol narration of this case is the following:

1. $A \rightarrow N_i : Can_i, Hi, \{K_{net}.Can_i\}_{K_{an_i}}, \{MAC(Can_i, Hi, \{K_{net}, Can_i\}_{K_{an_i}})\}_{K_{an_i}}$
2. $A \rightarrow N_i : Na, H1, \{P1, Na\}_{K_{net}}, \{MAC(Na, H1, \{P1, Na\}_{K_{net}})\}_{K_{net}}$

where $K_{an_i} := F(F(K_n, N_i), A)$ and $Na := A.Epoch.Ctr$.

In the first protocol narration message, the broadcast message sender A generates a new cluster key K_{net} . It sends it to all the intended receivers of the broadcast message. The cluster key is encrypted with the shared key of the node A and its neighbour. So, in our model, A sends this message as many times as

neighbours it has. Once all the intended message recipients know the cluster key, the node A sends the broadcast message. This message is sent once, but all the node A neighbours receive a copy. Broadcast messages are often used to update node configuration or internal data. So, the node A does not expect an answer to its broadcast message.

The fig. 5 represents the diagram transitions for the node A (sender node) and the node B (receiver node).

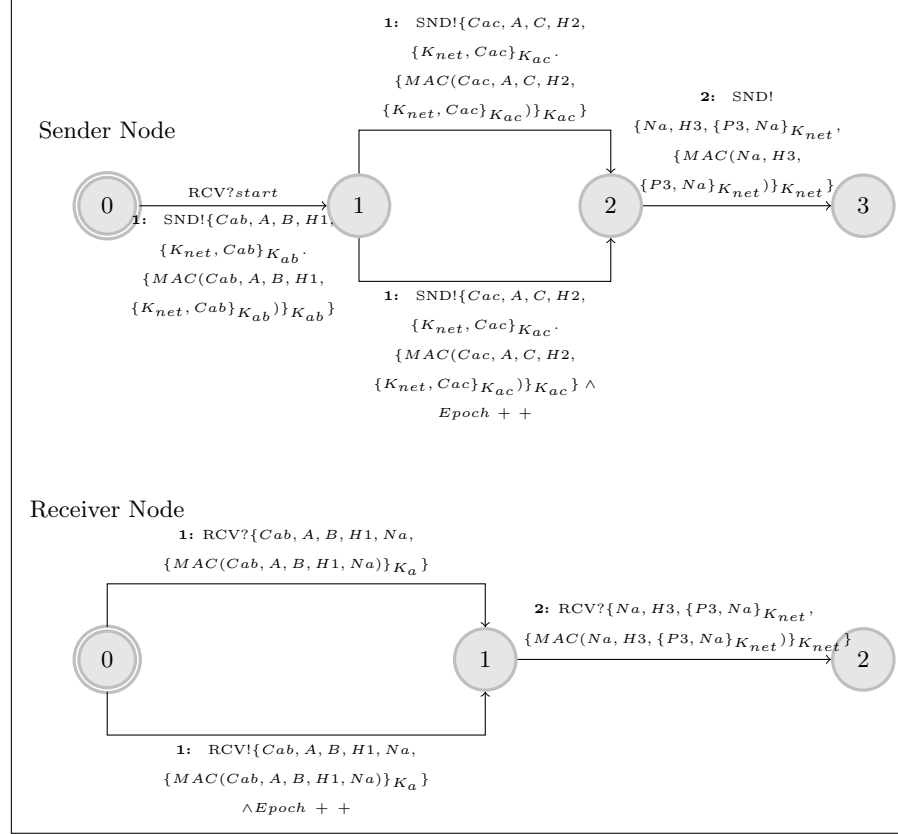


Fig. 5. MiniSec-B model diagram

The properties we have to analyse are the following:

- Authentication of $H1$ and $P1$, i.e., the node A , the node B and the node C share the same value for $H1$ and $P1$ and all of them execute the same session of the protocol. This property allows us to prove that there are not replay attacks and the freshness of messages is guaranteed.
- Confidentiality of $P1$, i.e., $P1$ is a secret value shared between A , B and C , and they are not known by an intruder or third parties.

We consider two executions of the protocol where there is a sender node A and two receiver nodes B and C . We also analyse a scenario where one of the receiver nodes is the intruder. AVISPA does not report about any attack in any of the cases.

5 Conclusions

In this paper we have presented a formal approach to the security analysis of MiniSec by means of a model checking tool called Avispa. We have analysed two cases.

The first case analyses the communication between two nodes with MiniSec-U mode. This case shows how a node can secure a request with MiniSec-U. First, the sender node establish LEAP pair of shared keys with the intended message recipient. Afterwards, it uses these keys to send a request and receive a response. We have not found any attack on the protocol. We can conclude is a secure solution for key distribution under our assumptions.

The second case represents MiniSec-B mode. This protocol allows a node to broadcast a message in a secure way. For that purpose, the sender establish a LEAP cluster key among the broadcast message receivers. We have not found any attack on the protocol. We can conclude is a secure solution for key distribution under our assumptions.

Our future work is concerned with extending our analysis to other security protocols for wireless sensor networks such as TinySec[14], μ TESLA (defined in [21]), and ZigBee[16]. Also we are interested in the analysis of TinySec with other distribution key protocols such as LEAP+[27] and TinyPK [25]. On the other hand, we are interested in the development of a more accurate intruder model and model checking tool that take into account the features of wireless networks, such as its constrained bandwidth and node mobility.

References

1. A. Armando, D. A. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. H. Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA tool for the automated validation of internet security protocols and applications. In K. Etessami and S. K. Rajamani, editors, *CAV*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285. Springer, 2005.
2. A. Armando and L. Compagna. An optimized intruder model for sat-based model-checking of security protocols. In A. Armando and L. Viganò, editors, *Electronic Notes in Theoretical Computer Science*, volume 125, pages 91–108. Elsevier Science Publishers, March 2005.
3. M. Backes, S. Mödersheim, B. Pfitzmann, and L. Viganò. Symbolic and cryptographic analysis of the secure WS-ReliableMessaging scenario. In L. Aceto and A. Ingólfssdóttir, editors, *FoSSaCS*, volume 3921 of *Lecture Notes in Computer Science*, pages 428–445. Springer, 2006.

4. D. A. Basin, S. Mödersheim, and L. Viganò. Ofmc: A symbolic model checker for security protocols. *Int. J. Inf. Sec.*, 4(3):181–208, 2005.
5. K. Bhargavan, C. Fournet, and A. D. Gordon. Verifying policy-based security for web services. In V. Atluri, B. Pfitzmann, and P. D. McDaniel, editors, *ACM Conference on Computer and Communications Security*, pages 268–277. ACM, 2004.
6. B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.
7. H. Chan, A. Perrig, and D. X. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, page 197. IEEE Computer Society, 2003.
8. Y. Chevalier, L. Compagna, J. Cuéllar, P. H. Drielsma, J. Mantovani, S. Mödersheim, and L. Vigneron. A high level protocol specification language for industrial security-sensitive protocols. In *Proceedings of Workshop on Specification and Automated Processing of Security Requirements (SAPS)*, pages 193–205, 2004.
9. Y. Chevalier and L. Vigneron. Strategy for Verifying Security Protocols with Unbounded Message Size. *Journal of Automated Software Engineering*, 11(2):141–166, April 2004.
10. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, 1999.
11. D. Dolev and A. C.-C. Yao. On the security of public key protocols. In *FOCS*, pages 350–357. IEEE, 1981.
12. L. Eschenauer and V. Gligor. A key-management scheme for distributed sensor networks. In V. Atluri, editor, *ACM Conference on Computer and Communications Security*, pages 41–47. ACM, 2002.
13. Y. Glouche, T. Genet, O. Heen, and O. Courtaf. A security protocol animator tool for AVISPA. In *ARTIST2 Workshop on Security Specification and Verification of Embedded Systems*, Pisa, May 2006.
14. C. Karlof, N. Sastry, and D. Wagner. 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*, pages 162–175. ACM, 2004.
15. I.-G. Kim and J.-Y. Choi. Formal verification of pap and eap-md5 protocols in wireless networks: Fdr model checking. *aima*, 02:264, 2004.
16. P. Kinney. Zigbee technology: Wireless control that simply works. www.zigbee.org/resources, Oct. 2003. Whitepaper.
17. G. Lowe. Casper: A compiler for the analysis of security protocols. *Journal of Computer Security*, 6(1-2):53–84, 1998.
18. M. Luk, G. Mezzour, A. Perrig, and V. Gligor. Minisec: A secure sensor network communication architecture. In *Proceedings of IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, April 2007.
19. J. C. Mitchell. Finite-state analysis of security protocols. In A. J. Hu and M. Y. Vardi, editors, *CAV*, volume 1427 of *Lecture Notes in Computer Science*, pages 71–76. Springer, 1998.
20. A. Perrig, J. A. Stankovic, and D. Wagner. Security in wireless sensor networks. *Commun. ACM*, 47(6):53–57, 2004.
21. A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler. SPINS: Security protocols for sensor networks. *Wireless Networks*, 8(5):521–534, 2002.
22. P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: a block-cipher mode of operation for efficient authenticated encryption. In *ACM Conference on Computer and Communications Security*, pages 196–205, 2001.

23. M. L. Tobarra, D. Cazorla, F. Cuartero, and G. Diaz. Application of formal methods to the analysis of web services security. In M. Bravetti, L. Kloul, and G. Zavattaro, editors, *EPEW/WS-FM*, volume 3670 of *Lecture Notes in Computer Science*, pages 215–229. Springer, 2005.
24. M. L. Tobarra, D. Cazorla, F. Cuartero, and G. Diaz. Formal verification of TLS handshake and extensions for wireless networks. In *Proc. of IADIS International Conference on Applied Computing (AC'06)*, pages 57–64, San Sebastian, Spain, February 2006. IADIS Press.
25. R. J. Watro, D. Kong, S.-f. Cuti, C. Gardiner, C. Lynn, and P. Kruus. TinyPK: securing sensor networks with public key technology. In S. Setia and V. Swarup, editors, *SASN*, pages 59–64. ACM, 2004.
26. S. Zhu, S. Setia, and S. Jajodia. LEAP: efficient security mechanisms for large-scale distributed sensor networks. In S. Jajodia, V. Atluri, and T. Jaeger, editors, *ACM Conference on Computer and Communications Security*, pages 62–72. ACM, 2003.
27. S. Zhu, S. Setia, and S. Jajodia. LEAP : Efficient security mechanisms for large-scale distributed sensor networks. *ACM Transactions on Sensor Networks*, 2(4):500–528, 2006.