Multihop Node Authentication Mechanisms for Wireless Sensor Networks*

Ismail Mansour[1,2], Damian Rusinek[3], Gérard Chalhoub[1,2], Pascal Lafourcade[1,2], and Bogdan Ksiezopolski[3,4]

[1] Clermont Université, Université d'Auvergne, LIMOS, BP 10448, F-63000, Clermont-Ferrand, France
[2] CNRS, UMR 6158, LIMOS, F-63173 Aubiuère, France
[3] Institute of Computer Science, Maria Curie-Sklodowska University Lublin, Poland
[4] Polish-Japanese Institute of Information Technology, Warsaw, Poland

Abstract. Designing secure authentication mechanisms in wireless sensor networks in order to associate a node to a secure network is not an easy task due to the limitations of this type of networks. In this paper, we propose different multihop node authentication protocols for wireless sensor networks. For each protocol, we provide a formal proof using Scyther to verify the security of our proposals. We also provide implementation results in terms of execution time consumption obtained by real measurements on TelosB motes. These protocols offer different levels of quality of protection depending on the design of the protocol itself. Finally, we evaluate the overhead of protection of each solution, using AQoPA tool, by varying the security parameters and studying the effect on execution time overhead of each protocol for several network sizes.

Keywords: Authentication, Wireless Sensor Network, Security, Quality of Protection, Multihop, Formal Verification.

1 Introduction

Wireless sensor networks (WSN) are more and more used in critical applications where the identity of each communicating entity should be authenticated before exchanging data in the network. The wireless nature of this technology makes it easy for intruders to try to intervene in the network activity and create any of the known attacks in WSNs [11]. Many of the current propositions focus on message authentication for ensuring data authentication and integrity, and some focus on user authentication to give access to the network for certain previously declared users. In this paper we propose a variation of different node authentication protocols that help authenticate any node in the network regardless of users.

Designing secure protocols is an error-prone task. One of the well known examples is the famous flaw found on the Needham Scroeder protocol seventeen years after its publication [19]. It clearly shows that designing secure protocols is not an easy task. During the last decades, several automatic tools for verifying the security of cryptographic protocols have been elaborated by several authors, like for instance Proverif [3], Avispa [25] or Scyther [4]. These symbolic tools use the Dolev-Yao intruder model [8], that considers that the intruder is controlling the network and makes the perfect encryption hypothesis[1]. The state of the art shows that formal methods are now mature and efficient enough to be used in the design of security protocol in order to avoid such logical flaws.

Another aspect which should be taken into account during WSN protocols analysis is performance which refers to the security operations. The traditional approach assumes that the best way is to apply the strongest possible security measures which make the system as secure as possible. Unfortunately,

* This reasearch was conducted with the support of the "Digital Trust" Chair from the University of Auvergne Foundation.

[1] Meaning that it is possible to obtain the plain text of an encrypted message only if the secret key is known.

such reasoning leads to the overestimation of security measures which causes an unreasonable increase in the system load [14]. The system performance is especially important in the systems with limited resources such as wireless sensor networks or mobile devices. The solution may be to determine the required level of the protection and adjust some security measures according to these requirements. Such an approach can be achieved by means of the Quality of Protection [12,13,15] where the security measures are evaluated according to their influence on the system security.

Contributions

The originality of our work resides in the fact that it combines several aspects of security, from designing secure protocols to evaluating the implementation of our solution, going through formal automatic analysis of security and quality of protection analysis. Our contributions can be summarized in the four following points:

1. Design of multihop node authentication mechanisms.
2. Formal automatic analysis of our solutions.
3. Implementation on TelosB motes.
4. Evaluation of the quality of protection of our solutions.

Our main contribution is the design of several secure authentication protocols. In order to avoid flaws, we use Scyther [23] to prove the correctness of all our protocols automatically. We have implemented our protocols on TelosB motes in order to obtain time consumption for few nodes. From the quality of protection analysis point of view, Scyther abstracts the cost of the communication and also does not consider the computation time of cryptographic primitives. The quality of protection analysis for WSN cryptographic protocols is almost impossible to perform manually. This increases the difficulty to design secure and efficient protocols at the same time. Using our real implementation on TelosB motes, we have designed several metrics to calibrate the Automated Quality of Protection Analysis tool (AQoPA[2]). With this tool we have evaluated the quality of protection of our protocols. This analysis takes into account all security factors which affect the overall system security to determine the fastest protocol according to the level of protection that is desired by the application.

Related Work:

Authentication protocols in multihop WSNs: Very few work has been done for node authentication protocols in multihop WSNs. Most of the existing authentication protocols proposed for WSNs neglect the multihop factor. In [1], authors proposed a protocol where the base station broadcast authentication elements for in range sensor nodes to be able to authenticate new arriving nodes. In fact, they consider that any previously authenticated node can authenticate new nodes.

In [7] and [28], authors propose an authentication mechanism for users and consider that sensor nodes inside the WSN are trusted nodes. In [28], authors propose a stronger authentication protocol that ensures mutual authentication and protection against attacks from other users, which is not the case for [7].

Recently in [9], authors propose an authentication model that aims at reducing overhead for the re-authentication of sensor nodes. It is based on a ticket encrypted using a common secret key between neighbouring fixed nodes. This ticket is sent to a mobile node during the first authentication phase. This ticket is only useful when the mobile node decides to re-authenticate with this neighbour fixed node. In addition, the protocol only works well when the fixed node is in direct range with the base station, the initial authentication phase suffers from internal attacks as other sinks in the network can easily take the place of one another when they are not in communication range with the base station.

In [29], authors propose a node authentication protocol for hierarchical WSNs. The hierarchical topology is limited to a base station, cluster heads and sensor nodes. The cluster heads can reach the sensors of their clusters directly, and can also reach the base station directly. The authentication is based on hash chain functions. The proposed protocol is not resilient to insider attacks as cluster

[2] AQoPA is available at: **http://www.qopml.org**.

heads are trusted to forward join requests to base station. In addition, the authors did not specify how the protocol copes with a multihop topology between cluster heads and the base station.
In our proposition, we take into account the multihop factor where any node in the network is able to be authenticated by sending a request in a multihop manner towards the base station. We also consider different cases depending on the level of trust we have in intermediate nodes and their computation capacities. Finally we formally prove the security using the automatic verification tool Scyther [4].

Quality of protection evaluation: In the literature several quality of protection models were created for different purposes and have different features and limitations. Authors in [17] attempted to extend the security layers in a few quality of service architectures. Unfortunately, the descriptions of the methods are limited to the confidentiality of the data and are based on different configurations of the cryptographic modules. In [27], authors created quality of protection models based on the vulnerability analysis which is represented by the attack trees. The leaves of the trees are described by means of the special metrics of security. These metrics are used for describing individual characteristics of the attack. In [13], authors introduced mechanisms for adaptable security which can be used for all security services. In this model the quality of protection depends on the risk level of the analyzed processes. Authors in [20] present the quality of protection analysis for the IP Multimedia Systems (IMS). This approach presents the IMS performance evaluation using Queuing Networks and Stochastic Petri Nets. In [16], authors create the adversary-driven, state-based system security evaluation. This method quantitatively evaluates the strength of the security of the system. In [24], authors present the performance analysis of security aspects in the UML models. This approach takes as an input a UML model of the system designed by the UMLsec extension [10]. This UML model is annotated with the standard UML Profile for schedulability, performance and time, and then analysed for performance.
In [12], the Quality of Protection Modelling Language (QoP-ML) is introduced. It provides the modelling language for making abstraction of cryptographic protocols that put emphasis on the details concerning quality of protection. The intended use of QoP-ML is to represent the series of steps which are described as a cryptographic protocol. During the analysis one cannot consider only primary cryptographic operations or basic communication steps. The QoP-ML introduces the multilevel protocol analysis that extends the possibility of describing the state of the cryptographic protocol. The analysis involves the elements such as: cryptographic primitives, communication steps, information security management, key management, security policy management, legal compliance, implementation of the protocol and cryptographic algorithms as well as other factors that influence the system security. Every single operation defined by the QoP-ML is described by the security metrics which evaluate the impact of this operation on the security requirements of the system. The QoP-ML models can be automatically evaluated by the Automated Quality of Protection Analysis tool (AQoPA).

Outline: In the next section, we present five different protocols for establishing secure mutlihop communications. Then in Section 3, we use Scyther to formally prove the security of our solutions. In Section 4, we make a qualitative evaluation of our five protocols using AQoPA, before concluding the paper in the last section.

2 Multihop Authentication Protocols for WSN

We propose several protocols that allow a node to join a multihop WSN in a secure way. We distinguish two classes of protocols:

1. Direct Join to the Sink (DJS): a node joins directly through the sink.
2. Indirect Join to the Sink (IJS): a node joins the network through intermediate nodes in order to reach the sink.

We use public key Elliptic Curve Cryptography (ECC), using parameters secp160r1 and secp128r1 given by the Standards for Efficient Cryptography Group [26]. Our implementation of ECC on TelosB is based on TinyECC library [18]. More precisely we use Elliptic Curve Integrated Encryption

Scheme (ECIES) the public key encryption system proposed by Victor Shoup in 2001. For all symmetric encryptions we use an optimized implementation of AES [6] with a key of 128 bits proposed by [21].
Before deployment, each node N knows the public key $pk(S)$ of the sink S and also its own pair of private and public keys, denoted $(pk(N), sk(N))$ respectively. Based on ECC, we have that $pk(N) = sk(N) \times G$, where G is a generator point of the elliptic curve. Using this material, each node N can compute a shared key with the sink S using a variation of the Diffie-Hellman key exchange without interaction between the nodes, denoted $K_{DH}(N, S)$. These computations can be done by the sink and by all nodes before deployment in order to preserve their energy.

- The sink knows its own secret key $sk(S)$ and the public key $pk(N)$ of a node N. The sink computes $K_{DH}(N, S) = sk(S) \times pk(N)$.
- Node N multiplies his secret key $sk(N)$ by the public key of the sink $pk(S)$ to get $K_{DH}(N, S)$.

Both computations give the same shared key since:

$$K_{DH}(N,S) = sk(N) \times pk(S) = sk(N) \times (sk(S) \times G) = (sk(N) \times G) \times sk(S) = pk(N) \times sk(S)$$

Notations
In what follows, we use the following notations to describe exchanged messages in our protocols:

- I: a new node that initiates the protocol,
- R: a neighbour of node I,
- S: the sink of the network (also called base station),
- J_i: the i-th intermediate node between R and S,
- n_A: a nonce generated by node A,
- $\{x\}_k$: the encryption of message x with the symmetric or asymmetric key k,
- $pk(A)$: the public key of node A,
- $sk(A)$: the secret (private) key of node A,
- $K(I, S)$: the session key between I and S,
- NK: the symmetric network key between all nodes of the network randomly generated by S,
- $K_{DH}(N, S)$: the shared symmetric key between N and S using the Diffie-Hellman key exchange without interaction described above.

2.1 Direct Join to Sink : DJS_{orig}

The protocol DJS_{orig} is the original protocol presented in [22]. It allows new nodes in range of the sink to join the network directly. We present this protocol in Figure 1. The new node I sends a direct request to S in order to establish a session key with it. The node I begins the join process by computing the symmetric key $K_{DH}(I, S)$ with the sink S. Then, node I generates a nonce n_I and adds its identity in order to form the request $\{n_I, \mathrm{I}\}$. The request is encrypted with $K_{DH}(I, S)$ and sent to S. Upon reception, in order to decrypt the request, node S computes $K_{DH}(I, S)$ using I's identity provided by the routing protocol. Then, S verifies the identity of I [3] and generates a new session key $K(I, S)$. The join response contains n_I, the identity of S and the new symmetric session key. The response is encrypted using $pk(I)$ and is sent to I. Only I is able to decrypt the response with its secret key $sk(I)$. We note that n_I helps I to authenticate S.

2.2 Indirect Protocols to Join the Sink

In this section, we present four different protocols that allow a new node, out of range of S, to join the network. A new node can join the network through a neighbour node that is already authenticated in the network. The main differences between these protocols is the way the authentication of nodes between R and S is established and how messages are forwarded between them. In what follows, we describe each proposed protocol. In Table 1, we summarize the main differences between the proposed protocols.

[3] S checks if the identity of I belongs to the list of deployed nodes

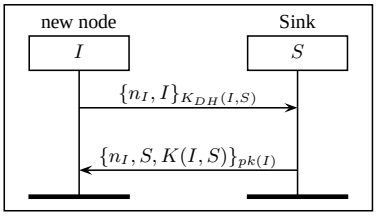


Fig. 1: DJS_{orig}: The node I joins directly the network by communicating directly with the sink S.

	Operations on intermediate nodes					
Protocol name	Authentication	Key type	from R to S		from S to R	
			Encrypt	Decrypt	Encrypt	Decrypt
IJS_{orig}	no	DH with S	no	no	no	no
$IJS_{NK,dec/enc}$	yes	network key	yes	yes	yes	yes
$IJS_{K,dec/enc}$	yes	session key	yes	yes	yes	yes
$IJS_{NK,onion}$	yes	network key	yes	no	no	yes

Table 1: Operations on intermediate nodes for the Indirect Join protocols.

The idea behind the different protocols is to allow the application to choose which protocol to use according to its constraints in terms of capacities, and needs in terms of security level. Using IJS_{orig} protocol is less consuming in terms of number of cryptographic operations but it assumes that all the nodes in the network are trusted nodes. $IJS_{NK,dec/enc}$ and $IJS_{K,dec/enc}$ protocols are similar in terms of number of operations but the latter is more resilient to node capture as it uses different keys along the route to the sink. As for $IJS_{NK,onion}$, it enables the network to do most of the cryptographic operations for the authentication process on the sink and thus reducing the computation time on intermediate nodes.

IJS_{orig}: This protocol is the original protocol presented in [22] and allows a new node to join the network through a neighbour node R. We present this protocol in Figure 2. The new node I sends an indirect request to S in order to establish a session key with R. The node R forwards the request to S through an intermediate nodes J_i. We note that the request and the response are just forwarded by J_i without any modifications. Node J_i is not able to decrypt any message due to the key used for encryption. Only nodes I and S are able to decrypt the messages encrypted with $K_{DH}(I, S)$, and only R and S are able to decrypt the messages encrypted with $K_{DH}(R, S)$.
In this protocol, the authors make the assumption that intermediate nodes are trusted. Hence, it is not resilient against insider attacks executed by intermediate nodes. Indeed an intruder can play the role of any intermediate node without being detected neither by the sink nor by the new node.
In what follows, we propose three protocols that allow a new node to join the network without trusting any intermediate node. Each solution uses a different approach for solving this question and has been proven secure using Scyther.

$IJS_{NK,dec/enc}$: The idea behind this protocol is to ensure authentication between all nodes by adding a nonce on each hop and by decrypting and encrypting exchanged messages as follows.
In Figure 3, we present $IJS_{NK,dec/enc}$ protocol. It allows new nodes to join the network through a neighbour node R using the network key for encryption/decryption on intermediate nodes. The node I sends a request containing a nonce with its own identity and the identity of R. Then, node R generates a nonce and adds it to the initial request before encrypting it with NK and forwarding it to J. Upon reception, node J decrypts the request and generates a new nonce n_J, adds it to the received

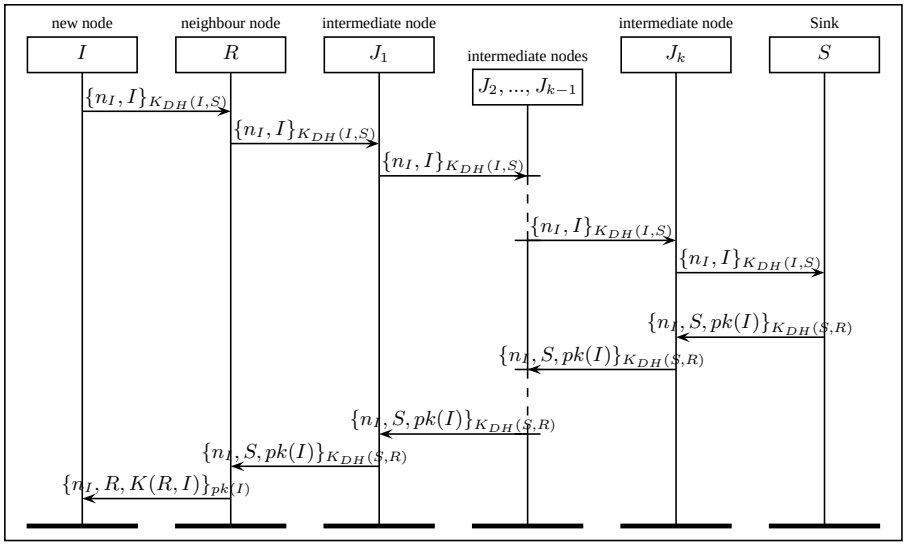


Fig. 2: IJS_{orig}: the original version. The intermediate node between R and S forwards messages without any encryption or decryption.

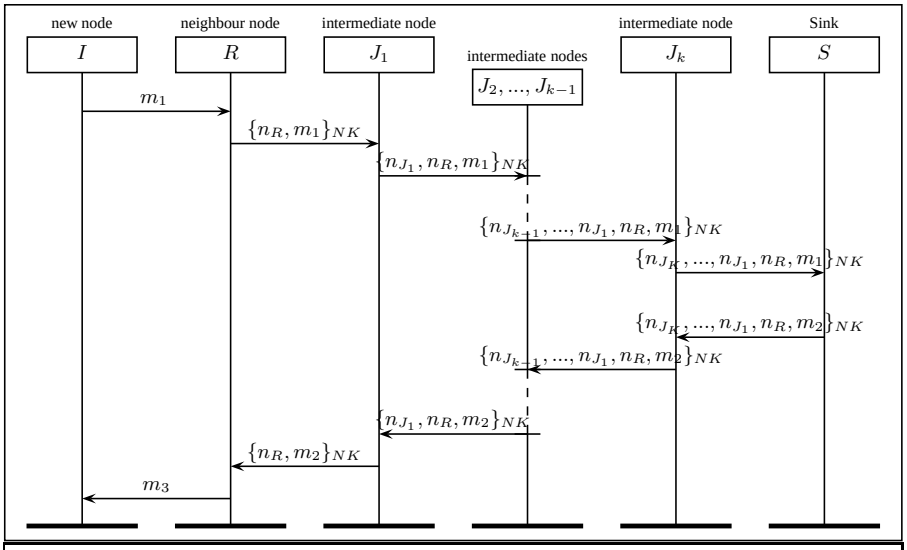
request and then encrypts the result using NK. When J receives the response message, it decrypts it using NK and extracts n_J, and then forwards the response message to R while keeping n_R in the message. We note that the nonce values n_I, n_R and n_J have helped S to authenticate I, R and J respectively and make sure that the request has been forwarded by previously authenticated nodes.
This protocol is secure as proven by Scyther [23], but each intermediate node has to decrypt and encrypt a message using the same key, which is the network key. Such cryptographic operations are very resources consuming. In addition, using the same key makes a node capture attack more dangerous for it enables the attacker to decrypt the authentication process of all nodes. In the next protocol, we avoid such risk by using a session keys.

$IJS_{K,dec/enc}$: In Figure 4, we present $IJS_{K,dec/enc}$ protocol. The two main differences between $IJS_{K,dec/enc}$ and $IJS_{NK,dec/enc}$ are:

- We encrypt and decrypt the request and the response between R and S with the symmetric session key $K(J_i, J_{i+1})$ established during the previous join phases.
- We also add all identities of intermediate nodes to the initial request sent by I.

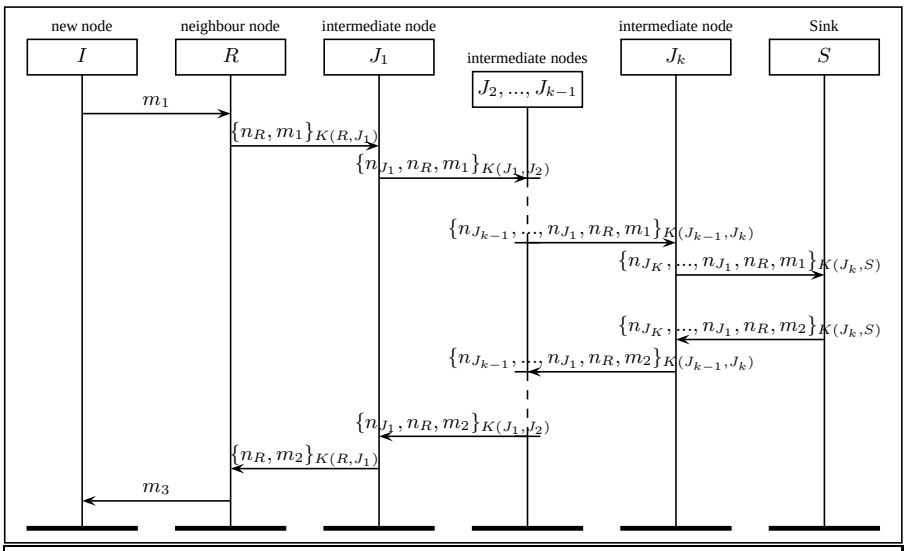
We assume that the node I is able to obtain the secure path to S from R. Indeed, the secure path is already known by R because it was able to join the network and build it using its routing protocol.
This protocol enhances the previous one by using session keys but still suffers from doing cryptographic operations on intermediate nodes. In the next protocol, we avoid overcharging intermediate nodes by doing most of the operations on the sink.

$IJS_{NK,onion}$: In Figure 5, we give a description of $IJS_{NK,onion}$ protocol which is an enhancement over $IJS_{NK,dec/enc}$ in terms of number of operations done by intermediate nodes. The goal is to help intermediate nodes to save time and energy. Using NK, an intermediate node J_i is able to add a nonce to the initial request and to encrypt the result before forwarding it. Upon reception, J_i is able to decrypt the response message, extract and retrieve its own nonce n_{J_i} and forward the rest of the message to R.
We note that the encryption/decryption operations that were not done by J_i are done by S. We assume that S is more efficient in computing and have more energy than the other nodes of the network.



$m_1 = \{n_I, I, R\}_{K_{DH}(I,S)}$
$m_2 = \{pk(I), \{n_I, I, R, S\}_{K_{DH}(I,S)}\}_{K_{DH}(R,S)}$
$m_3 = \{K(R,I), \{n_I, I, R, S\}_{K_{DH}(I,S)}\}_{pk(I)}$

Fig. 3: $IJS_{NK,dec/enc}$: The intermediate nodes J_i decrypt, add a nonce value and encrypt the result message before forwarding it. It uses the network key to encrypt/decrypt this messages.



$m_1 = \{n_I, I, R, J_1, ..., J_K\}_{K_{DH}(I,S)}$
$m_2 = \{pk(I), \{n_I, I, R, J_1, ..., J_K, S\}_{K_{DH}(I,S)}\}_{K_{DH}(R,S)}$
$m_3 = \{K(R,I), \{n_I, I, R, J_1, ..., J_K, S\}_{K_{DH}(I,S)}\}_{pk(I)}$

Fig. 4: $IJS_{K,dec/enc}$: The intermediate nodes J_i decrypt, add a nonce value and encrypt the result message before forwarding it. They use the session key to encrypt/decrypt this messages.

This protocol requires less computation for intermediate nodes, but suffers from exposure due to node capture attack exactly like $IJS_{NK,dec/enc}$ protocol because the same network key is used all the way from the source node to the sink.

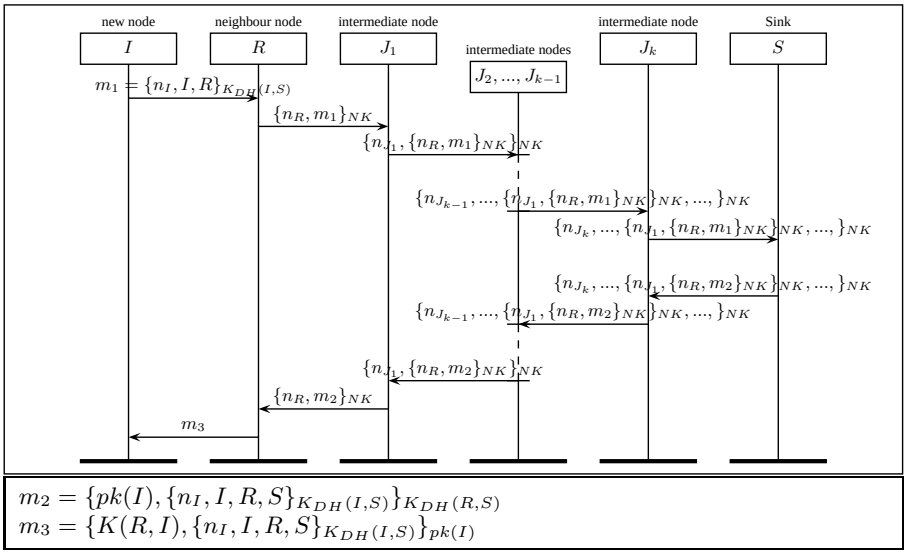


Fig. 5: $IJS_{NK,onion}$: The intermediate nodes J_i add a nonce and encrypt the request message and forward it to S.

3 Formal Security Evaluation

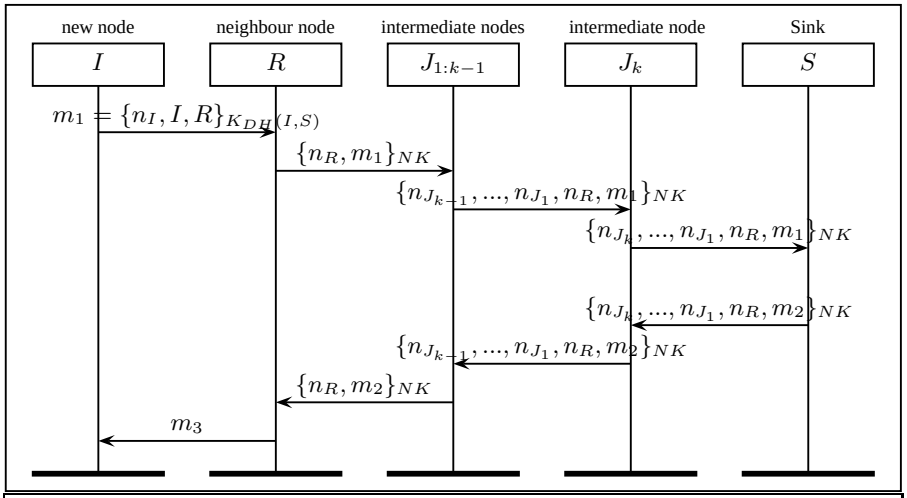
Evaluating the security of cryptographic protocols is not an easy task. It is easy to design flawed protocols. During the last decades several tools have been developed to automatically verify cryptographic protocols like for instance [2,3,4]. We use Scyther [4] because it is one of the fastest tools as it has been shown in [5] and one of the most user-friendly.

3.1 Scyther Overview

Cas Cremers has developed an automatic tool called Scyther [4]. It is a free tool available on all operating systems (Linux, Mac and Windows). This tool can automatically find attacks on cryptographic protocols and prove their security for bounded and unbounded numbers of sessions. One main advantage of Scyther is that it provides an easy way to model security properties like secrecy and authentication.

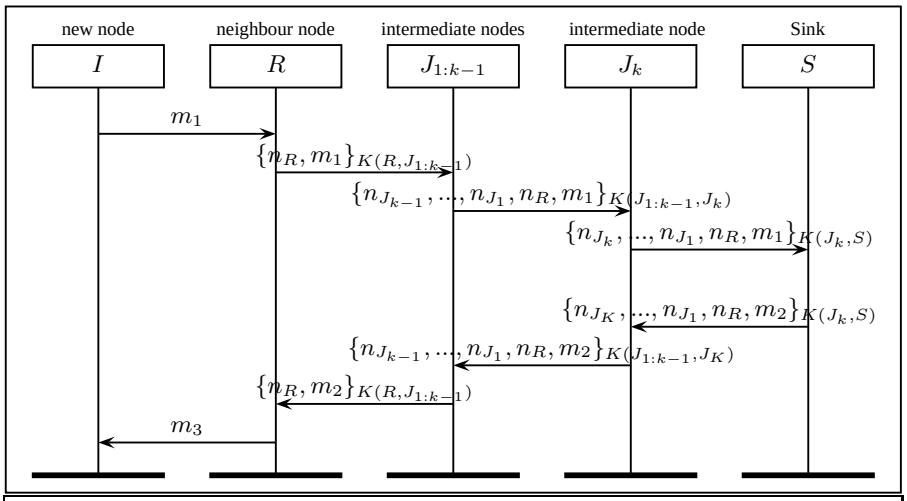
3.2 Results

We verified all our protocols using Scyther for a fix bounded number of participants. More precisely, we proved the secrecy of all sensitive data exchanged (keys and nonces) and also the authenticity of the communication. Our Scyther codes are available here [23] for more information.
Moreover, for all our protocols we proved by induction the security of the protocols for any number of intermediate nodes. Each time, the base case is proven using Scyther for a small number of nodes.



$m_2 = \{pk(I), \{n_I, I, R, S\}_{K_{DH}(I,S)}\}_{K_{DH}(R,S)}$
$m_3 = \{K(R,I), \{n_I, I, R, S\}_{K_{DH}(I,S)}\}_{pk(I)}$

Fig. 6: $IJS_{NK,dec/enc}$: Proof by Induction.



$m_1 = \{n_I, I, R, J_1, ..., J_k\}_{K_{DH}(I,S)}$
$m_2 = \{pk(I), \{n_I, I, R, J_1, ..., J_k, S\}_{K_{DH}(I,S)}\}_{K_{DH}(R,S)}$
$m_3 = \{K(R,I), \{n_I, I, R, J_1, ..., J_k, S\}_{K_{DH}(I,S)}\}_{pk(I)}$

Fig. 7: $IJS_{K,dec/enc}$: Proof by Induction.

– Protocol DJS: participants are one node and the sink. The verification using Scyther allows us to prove the security of our protocol.
– Protocol IJS_{orig}: Scyther found an authentication attack, where an intruder can replace any of the intermediate nodes between the new node and the sink and neither the sink nor the new node can detect its presence. This means that IJS_{orig} ensures only end-to-end authentication and fails to ensure hop-by-hop authentication. Hence, it is secure only if it is safe to send the join response through a route that was not the one used to send the join request. Indeed, in a hostile

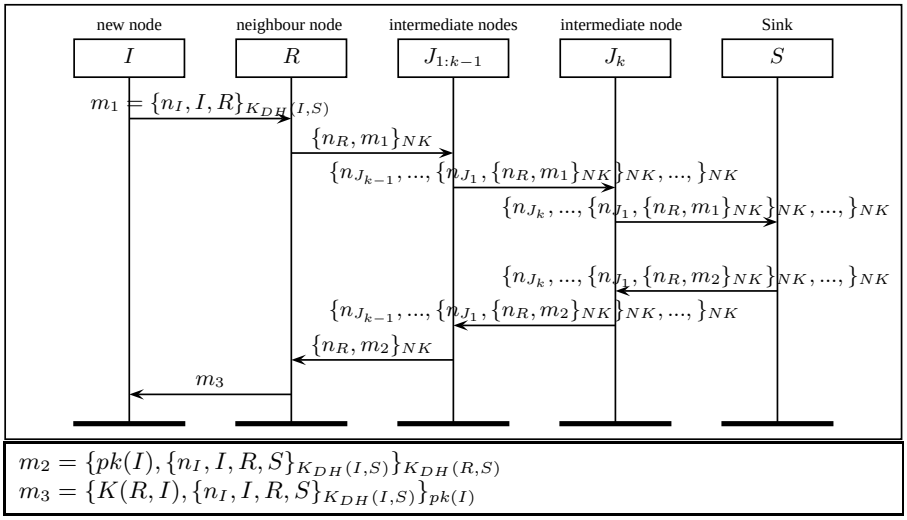


Fig. 8: $IJS_{NK,onion}$: Proof by Induction.

environment and in the presence of malicious nodes, it is important to be able to identify trusted nodes and be sure to route the response back through them in order to authenticate new nodes.

– $IJS_{NK,dec/enc}$, $IJS_{K,dec/enc}$ and $IJS_{NK,onion}$: these three protocols are constructed to work for any number of intermediate nodes, for each one of them we used the same method for proving their security. They ensure end-to-end and hop-by-hop authentication. In addition, we made a proof by induction. For the initialization of our induction, we used Scyther for proving that for 4 nodes all our protocols are secure. Then we assumed the protocols are secure for $k-1$ intermediate nodes, we showed that for k intermediate nodes they are still secure. Using the induction hypothesis, we obtain that the secrecy and authentication between I, R, J_1, ..., J_{k-1} and S is secure if S takes the place of J_k for all protocols. In order to prove the security when we add the intermediate node J_k, we consider the protocol between the following nodes I, R $J_{1:k-1}$, J_k and S (Figure 6, 7 and 8). Again using Scyther, we proved the security properties of these 5 nodes protocols.

This approach for generalizing the security of one protocol for an unbounded number of participants is a first step towards a new kind of protocols and also towards new security proofs. But it still remains a main challenge for the formal tool developers to elaborate new methods to perform such analysis automatically.

4 Quality of Protection evaluation

The differences in our protocols come from the usage of cryptographic primitives to ensure our security goals. We modelled our protocols using QoP-ML and we used AQoPA tool to analyse them. The model can be found in the QoP-ML models library (included in the AQoPA tool). For each protocol we examine two different scenarios with different key sizes for ECIES encryption and decryption. In the first scenario, we analysed the protocols with AES algorithm in CTR mode with a 128-bit key for symmetric encryption and ECIES for public key encryption with a 128-bit key. In the second scenario, we used a 160-bit key for ECIES. In the Table 2, we provide the real execution time for all our protocols for one intermediate node J, which means that we have the following 4 nodes: I, R, J, S. These results are the averages of 20 experiments of each scenario. We also give results of simulated execution time obtained with AQoPA tool. Notice that the time measurements slightly differ but remain within the standard deviation. This is due to the variations of execution time in the nodes

during the experiments. We used AQoPA tool in order to evaluate the overall overhead of security operations for each protocol for a large number of intermediate nodes in very big networks.

scenario 1 - ECIES - 128b key length					
Protocol name	Runtime of an actual time with S (ms)	Estimated time in AQoPA with S (ms)	Standard deviation (ms)	Estimated time in AQoPA without S (ms)	Gain %
JDS	9954.05	9920.00	123.14	3761.00	62%
JIS_{orig}	10127.32	10207.20	130.96	10071.20	1%
$JIS_{NK,dec/enc}$	10772.80	10823.16	127.40	10517.16	3%
$JIS_{K,dec/enc}$	10745.15	10823.88	125.26	10517.88	3%
$JIS_{NK,onion}$	10758.70	10823.16	126.56	10381.16	4%
scenario 2 - ECIES - 160b key length					
Protocol name	Runtime of an actual time with S (ms)	Estimated time in AQoPA with S (ms)	Standard deviation (ms)	Estimated time in AQoPA without S (ms)	Gain %
JDS	10102.35	10107.48	81.66	4113.48	60%
JIS_{orig}	10355.68	10396.60	109.13	10260.60	1%
$JIS_{NK,dec/enc}$	11072.75	11148.56	137.42	10808.56	3%
$JIS_{K,dec/enc}$	11069.20	11149.28	106.12	10809.28	3%
$JIS_{NK,onion}$	11043.05	11148.56	108.79	10638.56	4%

Table 2: Total times of joining new node with one intermediate node.

In Figure 9 (a), we present the execution time for all our protocols in both scenarios for 20, 40, 60, 80 and 100 intermediate nodes. Notice that the execution time for a key of 128 bits is almost equal to 160 bits. This is due to the fact that the code used is optimized for keys of 160 bits. The difference between the two scenarios become bigger when the number of intermediate node increases. Indeed, when the number of intermediate nodes increases, the number of cryptographic operations increases and the difference in execution time becomes bigger for bigger key sizes.

Note that the number of intermediate nodes gives roughly an idea about the radius of the network and not the size of the network. For example, when we evaluate a scenario with 20 intermediate nodes, it means that the furthest point of the network is 20 hops away from the sink. The total number of nodes in the network in that case will depend on the density of nodes. Keep in mind that simultaneous join request can be generated in the network and thus can take place at the same time.

It is important to notice how the time consumption of the original protocol is almost invariant when the number of intermediate nodes rises. Indeed, the main advantage of this protocol is that cryptographic operations are only done on the new node and the sink, intermediate nodes only forward the request and response without doing any additional cryptographic operation.

We also observe that $IJS_{NK,dec/enc}$ and IJS_{onion} protocols are more efficient than $IJS_{K,dec/enc}$. Indeed, for $IJS_{K,dec/enc}$ protocol, the join request has the list of all intermediate nodes starting from the first hop, whereas for $IJS_{NK,dec/enc}$ and IJS_{onion} each intermediate node adds its identifier as it forwards the requests. This makes the request message bigger for $IJS_{K,dec/enc}$ and thus needs more time for encryption and decryption along the route to the sink.

Moreover, the curves for IJS_{onion} and $IJS_{NK,dec/enc}$ are very close, because the same cryptographic operations are performed by different nodes. In order to compare them, in Figure 9 (b), we did not include the time consumption at the base station for all our protocols. As expected, the protocol IJS_{onion} is more efficient than the protocols $IJS_{NK,dec/enc}$ and $IJS_{K,dec/enc}$ for the global number of cryptographic operations is less important in intermediate nodes.

In Figure 10, we present the ratio of execution time of the sink over the total execution time of our protocols given in Figure 9 (a). We clearly see that IJS_{onion} is proposed for applications where sensor nodes are energy constrained but not the base station.

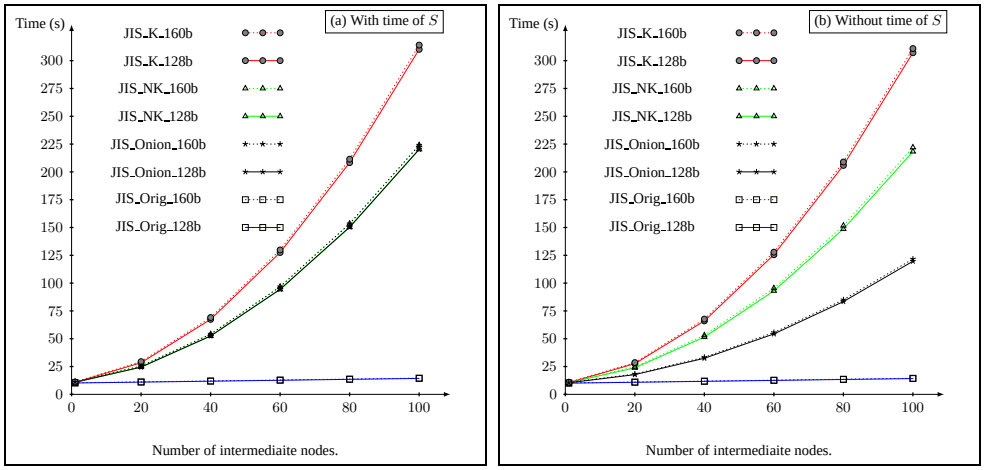


Fig. 9: Execution time of different protocols.

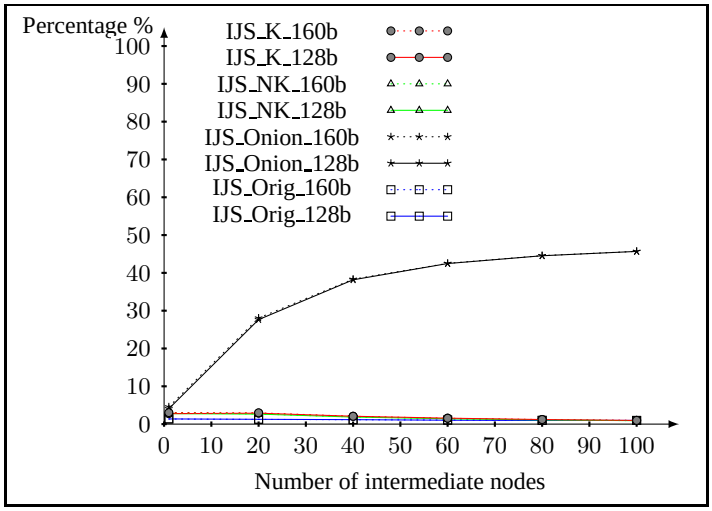


Fig. 10: Ratio of sink execution time over total execution time.

5 Conclusion and discussion

We proposed several multihop node authentication protocols for WSN. We proved the security of all of our solutions using the automatic tool Scyther. Moreover we implemented and tested all our protocols on TelosB nodes in order to evaluate the execution time of each of our solutions. Then we used AQoPA tool to perform an automatic evaluation of the overhead of protection of our solutions. Results show the cost in time consumption when the number of intermediate nodes separating the new node and the base station gets higher.

We studied different protocols that ensure different levels of security depending on the application needs. The original protocol supposes that the application does not need to use the same route for the join request and the join response. Indeed, in that case, all the nodes can participate in the routing operation for the authentication messages. This helped us significantly reduce the number of cryptographic operations. Only the new node and the sink are concerned by these operations which makes this proposal the most suitable one for very large multihop WSNs.

On the other hand, when dealing with more demanding applications, where the intermediate nodes are special nodes and have to be authenticated, more cryptographic operations are needed. We evaluated three protocols that respect that constraint. They differ, on one hand, in the resiliency against node capture attacks, and on the other, in the energy and calculation capacities assumption of the sink. With these protocols, the overhead of node authentication is very high, it reaches almost 5 minutes and 16 seconds in the most consuming scenario for 100 intermediate nodes. With the least consuming protocol, it takes around 2 minutes. Whereas the original takes around 15 seconds for authenticating a new node situated 100 hops away from the sink. The difference is significant and should be taken into account when we need to define the security needs.
We are currently working on the evaluation of key revocation and key renewal protocols for WSNs using Scyther and real testbeds on TelosB nodes. Key revocation and key renewal are very important mechanisms that need to be part of all security protocols. Our objective is to be able to achieve an acceptable security level for these protocols with the smallest number of cryptographic operations to limit the delay generated by the security overhead.

References

1. A. Al-mahmud and R. Akhtar. Secure sensor node authentication in wireless sensor networks. *International Journal of Computer Applications*, 46(4):10–17, May 2012. Published by Foundation of Computer Science, New York, USA.
2. A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. H. Drielsma, P.-C. Heám, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. R., J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA tool for the automated validation of internet security protocols and applications. In *Proc. of CAV'2005*, LNCS 3576, pages 281–285. Springer, 2005.
3. B. Blanchet. Automatic proof of strong secrecy for security protocols. In *IEEE Symposium on Security and Privacy*, pages 86–100, Oakland, California, May 2004.
4. C. Cremers. The Scyther Tool: Verification, falsification, and analysis of security protocols. In *Computer Aided Verification, 20th International Conference, CAV 2008, Proc.*, volume 5123/2008 of *Lecture Notes in Computer Science*, pages 414–418. Springer, 2008.
5. C. J. Cremers, P. Lafourcade, and P. Nadeau. Comparing state spaces in automatic protocol analysis. In *Formal to Practical Security*, volume 5458/2009, pages 70–94. Springer, 2009.
6. J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer-Verlag, 2002.
7. M. L. Das. Two-factor user authentication in wireless sensor networks. *IEEE Transactions on Wireless Communications*, 8(3):1086–1090, 2009.
8. D. Dolev and A. C. Yao. On the security of public key protocols. In *Proceedings of the 22Nd Annual Symposium on Foundations of Computer Science*, SFCS '81, pages 350–357, 1981.
9. K. Han and T. Shon. Sensor authentication in dynamic wireless sensor network environments. *International Journal of RFID Security and Cryptography*, 2012.
10. J. Jürjens. *Secure systems development with UML*. Springer, 2005.
11. T. Kavitha and D. Sridharan. Security vulnerabilities in wireless sensor networks: A survey. *Journal of Information Assurance and Security*, 5:31–34, 2010.
12. B. Ksiezopolski. QoP-ML: Quality of protection modelling language for cryptographic protocols. *Computers & Security*, 31(4):569–596, 2012.
13. B. Ksiezopolski and Z. Kotulski. Adaptable security mechanism for dynamic environments. *Computers & Security*, pages 246–255, 2007.
14. B. Ksiezopolski, Z. Kotulski, and P. Szalachowski. Adaptive approach to network security. In *Computer Networks*, volume 39 of *Communications in Computer and Information Science*, pages 233–241. Springer, 2009.
15. B. Ksiezopolski, Z. Kotulski, and P. Szalachowski. On qop method for ensuring availability of the goal of cryptographic protocols in the real-time systems. In *European Teletraffic Seminar 2011*, 2011.

16. E. LeMay, W. Unkenholz, D. Parks, C. Muehrcke, K. Keefe, and W. H. Sanders. Adversary-driven state-based system security evaluation. In *Proceedings of the 6th International Workshop on Security Measurements and Metrics*, MetriSec '10, pages 5:1–5:9. ACM, 2010.
17. S. Lindskog. *Modeling and Tuning Security from a Quality of Service Perspective*. PhD thesis, Chalmers University of Technology, 2005.
18. A. Liu and N. Ning. Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks. In *7th International Conference on Information Processing in Sensor Networks*, pages 245–256, April 2008.
19. G. Lowe. Breaking and fixing the needham-schroeder public-key protocol using fdr. *Software - Concepts and Tools*, 17(3):93–102, 1996.
20. A. Luo, C. Lin, K. Wang, L. Lei, and C. Liu. Quality of protection analysis and performance modeling in ip multimedia subsystem. *Comput. Commun.*, 32(11):1336–1345, July 2009.
21. N. Manica, M. Saloni, and P. Toldo. WSN - secure comunications with AES algoritms. University of Trento - Faculty of Computer Science, 2008.
22. I. Mansour, G. Chalhoub, and M. Misson. *Security architecture for multi-hop wireless sensor networks*. CRC Press Book, 2014.
23. I. Mansour and P. Lafourcade. Scyther code of our authentication protocols, dec 2013. `http://sancy.univ-bpclermont.fr/~lafourcade/scyther-code.tar`.
24. D. C. Petriu, C. M. Woodside, D. B. Petriu, J. Xu, T. Israr, G. Georg, R. France, J. M. Bieman, S. H. Houmb, and J. Jürjens. Performance analysis of security aspects in uml models. In *Proceedings of the 6th International Workshop on Software and Performance*, WOSP '07, pages 91–102. ACM, 2007.
25. V. B. Pérez, P. González, J. C. Cabaleiro, D. B. Heras, T. F. Pena, J. J. Pombo, and F. F. Rivera. Avispa: visualizing the performance prediction of parallel iterative solvers. *Future Generation Comp. Syst.*, 19(5):721–733, 2003.
26. C. Research. Standards for efficient cryptography, sec 1: Elliptic curve cryptography, September 2000.
27. Y. Sun and A. Kumar. Quality-of-protection (qop): A quantitative methodology to grade security services. In *ICDCS Workshops*, pages 394–399. IEEE Computer Society, 2008.
28. H.-L. Yeh, T.-H. Chen, P.-C. Liu, T.-H. Kim, and H.-W. Wei. A secured authentication protocol for wireless sensor networks using elliptic curves cryptography. *Sensors*, 11(5), 2011.
29. J. Zhang, R. Shankaran, M. A. Orgun, A. Sattar, and V. Varadharajan. A dynamic authentication scheme for hierarchical wireless sensor networks. In *MobiQuitous*, volume 73, pages 186–197. Springer, 2010.