PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 49, NO. 3–4, PP. 239–250 (2005)

SECURITY ANALYSIS OF SENSOR NETWORKS

Roland GÉMESI and László ZÖMBIK

Ericsson Hungary,
BME-TMIT,
e-mail:gemesi@alpha.tmit.bme.hu,laszlo.zombik@ericsson.com

Received: Sept. 11, 2004

Abstract

Wireless sensor networks distribute a common sensing and computing task within the large number of participants that use wireless communication. Such networks require a self-organizing and energy-aware set of protocols. Several protocols have beed designed for such environments, however to make certain proof of their secureness, their formal analysis is required.

In our article, we show an analysis framework capable of proving security properties of such protocols. Our methodology is based on the CSP process algebra. We will demonstrate its power by giving an attack possibility for an existing protocol, and the extensibility of the model will also be pointed.

Keywords: sensor networks, security analysis, CSP algebra.

1. Introduction

The development of compact, low-power wireless communication technologies evolves an entirely new kind of embedded systems. It has given an opportunity to have a large number of small-sized and heavily resource-constrained devices, which co-operatively serve a common goal. This distributed system requires a new set of communication protocols to provide reliable and secure operation. Although, several security protocols have been proposed, their security properties are truly guaranteed only after formal analysis. Analysis techniques that can deal with protocols of sensor networks are not well evolved yet.

In the rest of the introduction, wireless sensor networks and their special properties will be introduced and current analysis techniques will be shown. Section 2 gives an overview of the CSP (Communicating Sequential Processes) process algebra. Section 3 shows the CSP modelling aspects of security protocols being the base of our analysis. Section 4 shows some of our analysis performed and results are concluded in the last section.

1.1. Sensor Networks

Units of wireless sensor networks integrate sensors with CPU, memory, battery and wireless communication units [1]. This makes it possible to distribute sensing and

computing tasks. Units of sensor networks are usually heavily resource-constrained, therefore the amount of computing, storage and communication tasks should be minimized. The nodes are very small and unreliable components, therefore sensor networks require a self organizing and fault-tolerant architecture. Wireless networks that lack pre-built infrastructure are called mobile ad hoc networks.

Mobile ad hoc networks raise serious questions from the security point of view [2]. Their devices communicate with each other usually by using shared medium that is freely accessible even for malicious parties. Protection of confidential information should be achieved, although algorithms used in large computer networks are not usable in most cases. The limited resources do not allow the usage of hard cryptographic mechanisms, however information protected with only weak algorithms can be victims of brute force attacks. A self-organized environment makes authentication much more difficult, too. In traditional networks, authentication algorithms are mostly based on a trusted third party (Certificate Authority), which is not available in ad hoc networks. Moreover authentication algorithms are usually based on PKI (Public Key Infrastructure) that has high computational demands.

A number of security protocols had been proposed for sensor networks. They appear to be secure, however to make certain proof of it, formal analysis is required. The following subsection gives an overview of existing techniques creating such proofs.

1.2. Analysis of Security Protocols

The set of security claims against secure communication services have already been defined. Although self-organized systems introduce a new set of protocols built on different approaches, our security needs remain the same. In our work we focus on security, integrity, authentication and freshness.

A security protocol is a series of carefully designed messages between two or more participants. Their goal is to provide specified security properties for the participants even in the presence of an intruder. Security protocols rely on cryptographic operations to achieve their goals.

It is fundamental to define the capabilities of the attacker. The most widespread attacker model is the *Dolev-Yao* attacker model. It represents the fully *malicious medium*, since it can overhear, remove or even inject messages. It can build new messages even by performing deductions in its knowledge base. It has the constrain of *perfect cryptography*, that means that the used cryptographic primitives are perfect building blocks [3].

By now, a number of analysis techniques exist for the investigation of security protocols, such as the NRL protocol analyser, various process algebras or the strand spaces method [4, 5].

In the recent years, the CSP process algebra has been successfully used to introduce vulnerabilities of some widespread protocols. The method is fully automated and is capable of both proving security properties and also to introduce

counter examples of attacks. Although the method was designed to analyse key exchange protocols, the model is quite extensible. In the following section the process algebra used for our analysis will be introduced.

2. CSP Fundamentals

The CSP (Communicating Sequential Processes) process algebra was developed to describe and analyse concurrent systems. Concurrent systems are constructed from a number of independent components, which may interact with each other resulting a communication [6].

Describing a concurrent system in CSP is performed by specifying the behaviour of the participating *processes*. A process is defined in terms of the communication it can perform, known as atomic *events*. For example the P_1 process that performs event e and then becomes process Q is defined as $P_1 \mathrel{\widehat{=}} e \rightarrow Q$. The usage of compound events is a general concept in CSP. Event $c.e$ usually describes a channel c communicating event e. Event e can also be a complex event, thus the structure $c.e.f.g$ is also possible. Processes usually have parameters denoted by $P(par_1, \cdots, par_n)$, that makes it possible to represent several states of the same process.

The set of events known by a process is called its alphabet. The trace of a process is the set of its all possible event sequences. $STOP$ and $SKIP$ are the most trivial processes, the former produces no traces (visible events), while the latter expresses that a process has successfully terminated.

Processes can be combined via various operators to create a more complex process. $P_2 = Q \Box R$ expresses an *external choice*, which means that process P_2 can behave either as Q or as R. If Q and R offer different events to perform, P_2 can choose to perform any of them.

$P_3 \mathrel{\widehat{=}} Q \parallel_{\{E\}} R$ denotes that process P_3 is composed of Q and R with *parallelization* over the event set E. This means that events $e \in E$ of processes Q and R have to be performed simultaneously. This represents a system containing communication of its processes.

If there is no event, in which the Q and R processes interact, they are said to be independent. It can be described by the *interleave* operator denoted by $P_4 \mathrel{\widehat{=}} Q \mid\mid\mid R$.

Operators usually have an indexed version that expresses the usage of the given operator for each element of a set. For example, the *replicated external choice* $P_5 \mathrel{\widehat{=}} e \quad \{E\} \bullet e \rightarrow Q$ denotes that the external choice operator holds on each $e \in E$. Thus process P_5 can perform any event of the set and then becoming Q.

In the modelling of complex systems it is fundamental to describe building blocks that have invisible internal states and operation. The *hiding* operator $P_6 \mathrel{\widehat{=}} Q \setminus \{E\}$ denotes a process P_6 that behaves as process Q except that events $e \in E$ are not visible. It is possible to change the names of visible events with the *renaming*

operator. $P_7 \mathrel{\widehat{=}} Q\ e_1 \leftarrow e_2$ denotes process P_7 that behaves like process Q, however event e_1 is called e_2 any more.

With the help of these operators, it is possible to model systems built from distinct modules that are connected together through their interfaces and have their invisible internal operation.

Several models exist for the comparison of processes. In the rest of our paper the *refinement* checking of traces will be used. Process I *trace-refines* a process S, if and only if every trace of I is also a trace of S:

$$S \sqsubseteq_T I \iff traces(I) \subseteq traces(S)$$

In the traces model, checking such a refinement allows us to verify safety properties. We are able to say that a process (I) that represents an implementation of a system does not perform any unwanted behaviour that is not specified by a given specification (S) process. In the case of processes with a finite-state space, such analysis can be performed with the fully automated *FDR2* model checker.

3. Modelling and Analysis of Security Protocols

The CSP framework is especially suitable for the modelling of communication protocols [7]. Distinct participants pass messages of the protocol using their own rules.

Thus participants of a protocol are processes that use the $send$ and $recv$ channels for communicating with each other. Such events should also represent the source and destination agents and the message itself. Thus, for example the event $send.A.B.msg$ denotes that agent A sends message msg to agent B. This should be followed by $recv.A.B.msg$ when the destination agent B receives that message.

Events should be built from atomic elements, thus the message msg is also a construction of data variables and cryptographic operations. The most basic construction operator is the sequence of messages: $Sq.(m_1, m_2)$ denotes the sequence of m_1 and m_2. Encryption is constructed as $Encr.(data, key)$, that represents $data$ encrypted with key. Message authentication codes (MAC) will be also used in this paper. The construction $Mac.(key, data)$ results the authentication tag of $data$ generated by key.

Sending a message is usually well defined, since it is the role of the agent to create and send it. On the contrary, receiving a message is usually ambiguous, since it arrives from the environment and it has elements that can get several values. So, the reception of a message usually contains choices for each variants possibly being received. Thus, receiving a message is described using replicated external choice operator.

Participants are independent agents, that can be represented by applying the interleave operator for the agents. The system containing the behaviours of independent agents is given in the SYS_0 process. In this system, sending and receiving

messages are unordered, it can happen arbitrarily. The medium forces communication between the agents. In the case of the *Dolev-Yao* attacker model, it is the attacker itself that realizes the medium ($INTRUDER$). Therefore the messages are controlled by the unhonest medium, which is represented by the SYS process.

$$SYS_0 \widehat{=} ||| \ A \quad Agent \bullet A$$

$$SYS \mathrel{\widehat{=}} SYS_0 \parallel_{\{send,recv\}} INTRUDER$$

The process $INTRUDER$ is quite a complex process. It has an initial knowledge set and receives all messages. It has a number of deduction rules based on perfect cryptography, which makes it possible to analyse received messages and to synthesize new ones. Despite its internal complexity, its external behaviour is simply to interfere with the communication in any possible ways.

The composed system contains all possible event sequences of the protocol. The basis of our analysis is to check whether each of the possible states are safe. The set of appropriate safety properties should be formalized as a specification process, thus the $SPEC$ specification process should contain all the safe traces.

In many cases, only a part of the events are used for specification. Required events are renamed to a special $Signal$ channel and unneeded events are hidden from the system, while the $SPEC$ process will use only the $Signal$ channel. If the modelled system provides the specified security properties, the following refinement should hold:

$$SPEC \sqsubseteq_T SYS$$

The architecture of the model used for analysis can be seen in *Fig. 1*.

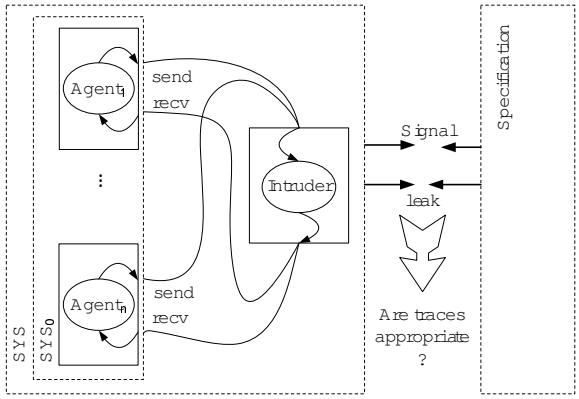

Fig. 1. The model of our analysis

3.1. Specification Process

The secret specification can be expressed as follos: if a node A claims that M is a secret with a node B then M must not be involved in the knowledge set of the intruder at all. Claiming such a secret is notified by a $Signal.ClaimSecret.A.M.B$ event. The intruder process indicates if the secret is acquired with the $leak.s$ event. Thus the specification process of secrecy can be formulated as follows [7] :

$$Secret\_Spec(M) \mathrel{\widehat{=}}$$
$$Signal.ClaimSecret?A\ M?B \rightarrow Secret\_Claimed(M))$$
$$\Box$$
$$leak.s \rightarrow Secret\_Spec(M)$$
$$Secret\_Claimed(M) \mathrel{\widehat{=}}$$
$$Signal.ClaimSecret?A\ M?B \rightarrow Secret\_Claimed(M)$$

The $Secret\_Spec(M)$ process allows event $leak.s$ arbitrarily, but the $ClaimSecret$ signal puts it to the $Secret\_Claimed$ state, where leakage is not allowed any more. It should be also decided which event of the original system should be renamed to the $ClaimSecret$ signal. In our work, we check secrecy as claimed only at the end of the protocol, thus the receiving of the last message should be renamed to the signal $ClaimSecret$.

The definition of the specification of authentication properties is the following. We say that A is *authenticated* to B means that if B thinks he has completed a run of the protocol with A, then A was also running the protocol with B [7]. Moreover, they should agree on the roles they performed, and in many cases they should also agree on some further data elements.

The following two signals are required:

$Signal.Running.Role_1.A.B.M$

$Signal.Commit.Role_2.B.A.M$

The first represents the fact that A believes it started running the protocol with B using data value M in the role named $Role_1$. The second signal is representing the fact that B believes he has committed a complete run of the protocol with A, and they agreed upon the value M in role $Role_2$.

Thus, the specification describes that a $Commit$ signal can occur only after a corresponding $Running$ signal. This can be defined as a simple process performing the two signals sequentially.

$$Auth\_Spec(B) \mathrel{\widehat{=}}$$
$$Signal.Running.Role_1.A?B?ms \rightarrow$$
$$Signal.Commit.Role_2.B.A.ms \rightarrow STOP$$

After performing appropriate renaming and hiding in the system, the desired property can be verified as a refinement assertion.

Although the introduced framework provides an automated facility to check security protocols, the creation of the model requires strong expertise. Since an adequate CSP model of a protocol usually takes more than 500 lines, the modelling is quite time-consuming and error-prone procedure. Fortunately it is possible to automate the CSP modelling of the protocol as well.

Casper is a compiler developed by LOWE [8] that helps the CSP modelling of security protocols, thus makes analysis much easier. Casper takes a more abstract description of the protocol and generates the corresponding CSP description. Its output file can be directly loaded into FDR, which makes the checking of refinement assertions. Casper also helps the interpretation of the results of the analysis, which is quite useful to understand an actual attack.

4. Analysis

Based on the framework introduced above, security analysis of several existing protocols will be presented. A possible attack will be introduced, and at the end of our analysis, it will be shown that the model is extensible to analyse secure ad hoc routing mechanisms.

4.1. Analysis of the SNEP protocol

In sensor networks, the protection of sensitive information should be provided with weak resources.

SNEP stands for *Sensor Network Encryption Protocol*, it is a protocol designed to provide the encryption needs of sensor networks. It applies only resource sparing cryptographic primitives such as symmetric key cryptography and one way hash function. It results very low communication and processing overhead and provides confidentiality, integrity, replay protection and freshness.

The protocol relies on previously shared keys between the communicating parties. It uses the key K_{enc} for encryption and K_{mac} for authentication purposes. Both of them are derived from a master key. Freshness is achieved by a counter C, and in the case of stronger requirements a nonce element is also used.

The protocol of our investigation was the following:

$$\begin{array}{llll} 1. & B & \rightarrow & A \qquad N_b, R_b \\ 2. & A & \rightarrow & B \qquad \{D\}\{K_{enc}, C\}, MAC_{K_{mac},C}(N_b, \{D\}\{K_{enc}\}) \end{array}$$

First, the base station B sends a R_b request tag together with the freshly generated N_b nonce. In reply, the sensor A sends the requested information D protected with the SNEP mechanism. First, the data D is encrypted using the encryption key K_{enc} and the counter C. A message authentication code (MAC) is also created for the encrypted data using the key K_{mac} and the counter C. The structure of the protected data can also be seen in *Fig. 2*.

The model generated by the Casper compiler is appropriate to perform the analysis. In the actual analysis the counter C was omitted, since the used N_b nonce provides much stronger freshness property. The specification of the protocol was the secrecy of the data D and its authenticity.

The CSP descriptions of the agents are the following:

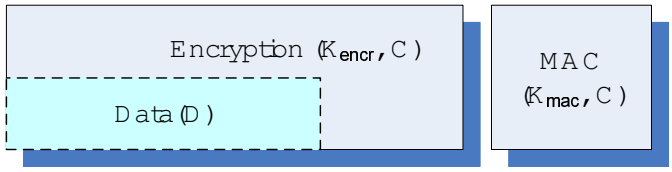

Fig. 2. SNEP protocol

$$
\begin{aligned}
&Base(B, N_b, R_b, K_{enc}, K_{mac}, A) = \\
&\quad send.B.A.Msg_1.Sq.(N_b, R_b) \rightarrow \\
&\qquad \Box D \quad Message\bullet \\
&\quad recv.A.B.Msg_2.Sq.(Encr.(D, K_{enc}), \\
&\qquad\qquad MAC.(K_{mac}, Sq.(N_b, Encr.(D, K_{enc})))) \rightarrow \\
&\quad SKIP \\
&Sensor(A, D, K_{enc}, K_{mac}, B) = \\
&\qquad \Box N_b \quad Nonce \bullet \Box R_b \quad Message\bullet \\
&\quad recv.B.A.Msg_1.Sq.(N_b, R_b) \rightarrow \\
&\quad send.A.B.Msg_2.Sq.(Encr.(D, K_{enc}), \\
&\qquad\qquad MAC.(K_{mac}, Sq.(N_b, Encr.(D, K_{enc})))) \rightarrow \\
&\quad SKIP
\end{aligned}
$$

The parameters of the agents are their identities followed by required knowledge such as nonces, keys and the other agent. The first event of the agent $Base$ is the $send$ event of message 1. Here the agent arrives at a choice, since the following receiving event $recv$ may contain any possible D data element protected by the mechanism. After receiving message 2, the base agent terminates successfully ($SKIP$). The $Sensor$ agent receives its first message by replicated choices, since it receives new elements. The next step is to send message 2 by event $send$. After this, the process terminates successfully.

Specifications of the protocol involve the secrecy of data D and the authenticity of both data and the participants. In the analysis of the protocol, no attack was found. The analysis was performed for several participants and protocol runs and, based on our analysis, the protocol can be stated to be secure.

With small modifications special cases were also analysed, such as the case of inappropriate nonces. The specification of a nonce is that it is a freshly generated unpredictable random number that can be applied only once. The sensor implementation of weak random number generator raises a threat, since our analysis has shown that in the case of an inappropriate nonce, a replay attack is possible.

Furthermore, the effects of key compromizes were also analysed, since sensors usually cannot have hard physical protection. If the intruder gets to know k_{enc}, confidentiality fails. The analysis has shown that in the case when k_{mac} is compromized, but k_{enc} remains valid, the protocol still holds the required properties. This is because k_{enc} protects the information in the case of perfect cryptography, thus faking the MAC does not raise any possible attacks. If both of them are known by the intruder, both specifications fail.

4.2. Analysis of a Sensor Key Exchange Protocol

Since resource-constrained sensor networks usually utilise only symmetric key cryptography, key sharing is fundamental for the bootstrapping of nodes. Traditional solutions usually apply public-key cryptography to share keys, but such a computationally expensive operation is not permitted in sensor networks.

SPINS proposes a Node-to-Node Key Agreement Scheme that is based on the SNEP mechanism [9]. The proposed scheme uses a base station as a trusted agent. Sensor networks usually have at least one such point for centralising measurement results. Thus, it is realistic to rely on such a level of infrastructure.

$$
\begin{array}{llcll}
1. & A & \rightarrow & B & N_A, A \\
2. & B & \rightarrow & S & N_A, N_B, A, B, MAC_{K_{BS}}(N_A, N_B, A, B) \\
3. & S & \rightarrow & A & \{SK_{AB}\}_{K_{AS}}, MAC_{K_{AS}}(N_A, B, \{SK_{AB}\}_{K_{AS}}) \\
4. & S & \rightarrow & B & \{SK_{AB}\}_{K_{BS}}, MAC_{K_{BS}}(N_B, A, \{SK_{AB}\}_{K_{BS}})
\end{array}
$$

The order of messages can be seen in *Fig. 3*. In the first message, node A sends its identifier together with a freshly generated unique N_A nonce. Node B turns to the server S with a request containing the received and a new N_B nonce, the agent identifiers and a MAC. This MAC is generated with the K_{BS} key, which is shared between B and S. The server receiving such a signed request sends the SK_{AB} session key to both of the participants authenticated in messages 3 and 4.

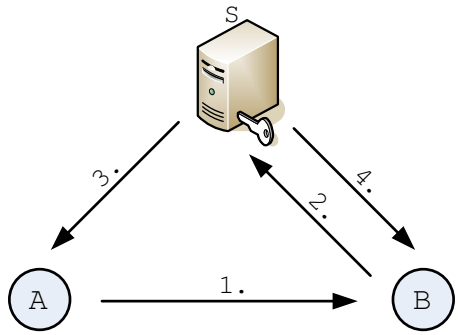

Fig. 3. Sensor key sharing protocol

Our analysis resulted the following counter example as an attack:

$$
\begin{array}{llcll}
\alpha 1. & I(B) & \rightarrow & A & N_M, B \\
\alpha 2. & A & \rightarrow & I(S) & N_M, N_A, B, A, MAC_{K_{AS}}(N_M, N_A, B, A) \\
\beta 1. & I(A) & \rightarrow & B & N_A, A \\
\beta 2. & B & \rightarrow & S & N_A, N_B, A, B, MAC_{K_{BS}}(N_A, N_B, A, B) \\
\beta 3. & S & \rightarrow & I(A) & \{SK_{AB}\}_{K_{AS}}, MAC_{K_{AS}}(N_A, B, \{SK_{AB}\}_{K_{AS}}) \\
\alpha 4. & I(S) & \rightarrow & A & \{SK_{AB}\}_{K_{AS}}, MAC_{K_{AS}}(N_A, B, \{SK_{AB}\}_{K_{AS}})
\end{array}
$$

As this description shows, the α run of the protocol misleads A. After the last message, A thinks that the SK_{AB} key is shared with B, however B was not involved in the α run at all. This flaw is realistic, since participants of a sensor network can both initiate a key sharing and also act as a responder.

The reason of the problem is that the last two messages do not identify both participants. To correct this flaw, the 3rd and 4th messages of the protocol should be extended with freshness information (nonces) of both participating agents.

After our extensions had been made, the analysis has shown the protocol to be secure. This procedure demonstrates the usage of our analysis framework for an iterative protocol development.

4.3. Analysis of the Ariadne Protocol

The Ariadne protocol was designed to provide secure routing for ad hoc networks. Routing mechanisms generally do not deal with the presence of unhonest participants. Therefore an unsecure routing protocol can be mislead in several ways [2].

The routing mechanism of Ariadne is based on the Dynamic Source Routing (DSR) protocol. The initiator broadcasts a route request message ($rreq$) that contains the source and destination addresses (S, D). A node (F_i) receiving such a message appends its own identifier to the request and rebroadcasts the message, therefore the request floods through the network. When the request arrives at its destination, a reply packet ($rrep$) is generated and sent back through the route specified in the packet.

DSR does not deal with the authentication of the participating nodes. When a malicious party is present in the network, authentication of protocol participants is required to exclude them from the communication. The Ariadne protocol provides authentic route discovery with the help of message authentication codes (MAC).

Communicating endpoints generate such authentication codes based on previously shared secret keys (k_{SD}). The authentication of unknown intermediate nodes is based on the delayed disclosure of keys. This means that a MAC is generated using a secret key. After the MAC is received by a specific agent, the key is disclosed. The authenticity of the message is provided by checking if the MAC was appropriate. In the Ariadne protocol, key disclosure is triggered by the route reply messages.

The messages of the protocol are the following, where newly generated message parts are emphasized.

$$
\begin{array}{lcll}
S & \rightarrow & * & \mathbf{rreq}(S, D, \langle\rangle), \mathbf{h_0}, \langle\rangle \\
F_i & \rightarrow & * & rreq(S, D, \langle F_1 \ldots F_{i-1}, \mathbf{F_i}\rangle), \mathbf{h_i}, \langle M_{F_1} \ldots M_{F_{i-1}}, \mathbf{M_{F_i}}\rangle \\
F_n & \rightarrow & D & rreq(S, D, \langle F_1 \ldots F_n\rangle), h_n, \langle M_{F_1} \ldots M_{F_n}\rangle \\
D & \rightarrow & F_n & \mathbf{rrep}(S, D, \langle F_1 \ldots F_n\rangle), \langle M_{F_1} \ldots M_{F_n}\rangle), \mathbf{M_D}, \langle\rangle \\
F_i & \rightarrow & F_{i-1} & rrep(S, D, \langle F_1 \ldots F_n\rangle), \langle M_{F_1} \ldots M_{F_n}\rangle), M_D, \langle k_n \ldots k_{i+1}, \mathbf{k_i}\rangle \\
F_1 & \rightarrow & S & rrep(S, D, \langle F_1 \ldots F_n\rangle), \langle M_{F_1} \ldots M_{F_n}\rangle), M_D, \langle k_n \ldots k_1\rangle
\end{array}
$$

The source node (S) initiates the request that contains the source and destination addresses, and an empty list to the identifiers of the intermediate nodes ($rreq(s, d, \langle\rangle)$). The source also generates a message authentication code (h_0) with the key that is shared with the destination ($h_0 = MAC_{k_{SD}}(rreq(s, d, \langle\rangle))$).

Intermediate nodes (F_i) append their addresses to the route list, which is received in the request ($\langle F_1 \ldots F_{i-1}\rangle$). Moreover, they create a new hash tag with the identifier and the received hash tag : $h_i = hash(F_i, h_{i-1})$. An authentication code is also created with an appropriate secret key from the whole message ($M_{F_i} = MAC_{k_i}(m)$). Authentication tags are collected in a list in the message, so the destination receives the authentication elements of each forwarding nodes, which are present in the route list ($\langle M_{F_1} \ldots M_{F_n}\rangle$).

The destination node gets the route request with all the n participating nodes ($\langle F_1 \ldots F_n\rangle$). It also appends a MAC, but it is based on the shared key with the source ($M_D = MAC_{k_{SD}}(m)$). It generates the reply message ($rrep$) and sends it back through the route with the authentication tags. When a message arrives to an intermediate node, it discloses the key used for authentication earlier (k_{F_i}).

By the end of this mechanism, the source node gets to know a possible route to the destination that contains only authenticated participating nodes as forwarding agents. This can be formalized as the authenticity of the identities of intermediate agents.

The number of intermediate nodes can vary ($F_1, \cdots, F_n$). Agents running the AODV protocol are similar. Forwarding and destination nodes have the same programme, but the node initiating the protocol is different. Thus two kinds of agents should be modelled: a general AODV and an initiator agent.

The construction of received and sent messages usually depends on the number of intermediate hops. For example a forwarder agent can receive several route request messages, which are constructed differently. The complexity of the received message specifies the agent's role in the protocol. Agents can even be extended with further choices to behave depending on the received message.

We have extended the model to check the Ariadne protocol this way. However the large number of choices results in a state space explosion, the case of allowing only one intermediate hop could be analysed and attack was not found. We have also set the case of two intermediate nodes, but the analysis requires more computing resources than we have. In some cases, data independence techniques can be applied to make the analysis shorter [11].

The main contribution of this analysis is to point out that the security of ad hoc routing protocols can also be analysed using the introduced framework. Systems involving a large number of similar agents can be analysed using CSP, since the used communication model is quite general and extensible.

5. Summary

We have presented that the protocols of sensor networks usually differ from traditional solutions. Proposed protocols should be analysed from the security point of view to proof their correctness or to show their flaws.

We have introduced an analysis framework based on the CSP process algebra that can be used to analyse security of distributed systems. We have built a model for the verification of sensor network security protocols.

The analysis framework was demonstrated by several examples. An attack of a proposed key exchange protocol was introduced and corrected. We have also presented that the model is extensible for routing protocols too.

References

[1] CULLER, D. – ESTRIN, D. – SRIVASTAVA, M., Overview of Sensor Networks. *IEEE Computer*, 2004/8, pp. 41–49.

[2] GÉMESI, R. – IVÁDY, B. – ZÖMBIK, L. Security of Mobile ad hoc Networks. *Híradástechnika*, 2002/12, pp. 2–8.

[3] DANNY, D.– YAO, A. C., On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, 1983, vol. IT-29, 12, pp. 198–208.

[4] MEADOWS, C., The NRL Protocol Analyzer: An Overview. *Journal of Logic Programming*, 1996/26/2, pp. 113–131.

[5] THAYER FÁBREGA, F. J. – HERZOG, J. C. – GUTTMAN, J. D., Honest ideals on Strand Spaces. Proceedings of the 11th IEEE Computer Security Foundations, 1998/6.

[6] STEVE SCHNEIDER, Concurrent and Real-time Systems. The CSP Approach. 2000.

[7] RYAN, P. – SCHNEIDER, S. – GOLDSMITH, M. – LOWE, G. – ROSCOE, B., Modelling and Analysis of Security Protocols, 2001.

[8] LOWE, G. – BROADFOOT, PH. – MEI LIN HUI. A Compiler for the Analysis of Security Protocols. *The 10th Computer Security Foundations Workshop*, 2001.

[9] ADRIAN PERRIG, ROBERT SZEWCZYK, VICTOR WEN, DAVID CULLER, J. D. TYGAR. SPINS: Security Protocols for Sensor Networks. Mobile Computing and Networking, 2001, pp. 189–199.

[10] YIH-CHUN HU, ADRIAN PERRIG, DAVID B. JOHNSON. ARIADNE: A Secure On-Demand Routing Protocol for Ad Hoc Networks. Mobicom, 2002. The 8th ACM International Conference on Mobile Computing and Networking

[11] BROADFOOT, P.J., Data Independence in the Model Checking of Security Protocols. University of Oxford, 2001