Tatra Mt. Math. Publ. **47** (2010), 81–97
DOI: 10.2478/v10127-010-0032-7

TATRA MOUNTAINS
Mathematical Publications

FORMAL ANALYSIS OF SECURITY PROTOCOLS FOR WIRELESS SENSOR NETWORKS

Marián Novotný

ABSTRACT. Design of security protocols is notoriously error-prone. For this reason, it is required to use formal methods to analyze their security properties. In the paper we present a formal analysis of the Canvas protocol. The Canvas protocol was developed by Harald Vogt and should provide data integrity in Wireless Sensor Networks. However, Dieter Gollmann published an attack on the protocol. We consider the fallacy of the Canvas scheme in different models of the attacker and present a solution for correcting the scheme. We propose a formal model of the fixed Canvas protocol in the applied pi-calculus. This model includes a model of the network topology, communication channels, captured nodes, and capabilities of the attacker. Moreover, we formulate and analyze the data integrity property of the scheme in the semantic model of the applied pi-calculus. We prove that the fixed Canvas scheme, in the presence of an active adversary, provides data integrity of messages assuming that captured nodes are not direct neighbors in the communication graph of a sensor network. Finally, we discuss the applicability of the proposed formal model for analysis of other WSN security protocols.

1. Introduction

The development of *Wireless Sensor Networks* (*WSNs*) was originally motivated by military applications. However, WSNs are now used in many civilian applications including environment monitoring, home automation, and traffic control. Recent advances in electronics and wireless technologies have enabled the development of large scale sensor networks which consist of many low-power, low-cost, and small-sized sensor nodes.

According to the above mentioned applications of sensor networks, it is important to ensure security properties such as *confidentiality, authenticity*, and *data integrity*. Traditional security techniques such as protocols for *key establishment* and *authentication* cannot be applied directly, because sensor devices are limited in their computation, energy, and communication capabilities. Moreover,

2010 Mathematics Subject Classification: 94A60, 68Q60.
Keywords: security protocols, Wireless Sensor Networks, formal analysis, Dolev-Yao model of the attacker, data integrity, the applied pi-calculus.
This work was supported by APVV grant SK-SI-0010-08 and Slovak VEGA grant 1/0035/09.

they are often deployed in accessible areas, where *capture* by an attacker is possible. In literature, a sensor device is not considered *tamper-resistant.* Making all devices of a sensor network tamper-proof would be impossible in general due to increased costs. This way, the attacker can copy all secrets from a device and fully determine its operation. Furthermore, the typical communication in WSNs is not *point-to-point*, but *one-to-many*, or *many-to-one.* Therefore, standard security solutions such as SSH or SSL protocols cannot be applied directly and the design of security protocols for WSNs has become a challenge in the computer security research field. Many protocols originally designed for WSNs have been developed [12]. We focus on the *Canvas scheme* [11] which should provide data integrity in a sensor network.

Since design of security protocols is notoriously error-prone, it is necessary to model and analyze their security properties formally. The seminal work on formal analysis of WSN security protocols was done in the paper [8]. The authors used the *model checking* tool *AVISPA* [3] and modeled and analyzed *TinySec* [12] combined with *LEAP* [12] protocol in various communication scenarios. They analyzed the protocols in the standard *Dolev-Yao model* of the attacker. In this model, the communication network is described vaguely. On the other hand, in order to analyze WSN security protocols we need a formal model which adequately expresses the network topology and communication channels. Developing a more detailed model of the attacker was formulated as a challenge in the computer security research field [6]. We propose a formal model of a fixed Canvas scheme in the applied pi-calculus, which includes a model of the network topology, communication channels, captured nodes, and capabilities of the attacker. Moreover, we show a technique how to formulate and prove security properties of the Canvas scheme in the proposed formal model.

This paper is organized as follows. The next section describes the Canvas protocol. The section following next discusses various models of the attacker with respect to capabilities of the attacker. This section also depicts a security fallacy of the Canvas protocol and provides a solution for correcting the scheme. In Section 4 we present a formal model of the fixed Canvas protocol in the applied pi-calculus. In Section 5 we formulate and analyze the data integrity property of the scheme in the semantic model of the applied pi-calculus. The last section presents our conclusions and discusses the applicability of the proposed formal model for analysis of other WSN security protocols.

2. The Canvas protocol

A sensor network can be represented by a *communication graph* $G = (V, E)$, where *nodes* from V represent sensor devices and *edges* from $E \subseteq V \times V$ established communication links. A *path* in a graph G is a sequence of distinct nodes from V, such that from each of its nodes there is an edge from E to the next node in the sequence. A *length of a path* is the number of edges in the path. Let $\pi^l(G)$ denote the set of all paths of the length l in a graph G. We define a *set of i-hop neighbors* $N_i(v)$ of a node $v \in V$ as the set of nodes $u \in V$ such that there exists a path $\rho \in \pi^i(G)$ starting in v and ending in u.

The Canvas protocol was designed by H. V o g t and published in [9], [10], [11]. It should provide *data integrity* or *data origin authentication* [7] in a sensor network under the assumption that captured nodes are *isolated*, i.e., captured nodes are not direct neighbors in a communication graph. We also assume that there exists an algorithm for routing a message in the network. Each node shares secret keys with nodes that are one or two hops away as shown in Fig. 1.

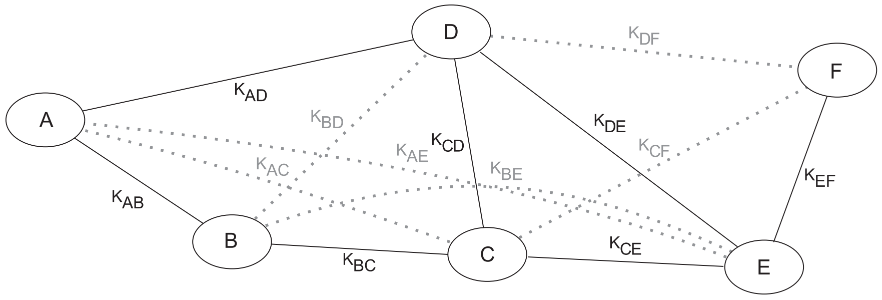

FIGURE 1. A simple sensor network with established keys in the Canvas scheme.

Message authentication codes—MACs [7] are used to protect the integrity of a message between nodes which share a secret key. Let K_{AB} denote the key shared between the nodes A, B and $h(m, K_{AB})$ denote the MAC of a message m using the shared key K_{AB}. For the sake of simplicity, we demonstrate the functionality of the Canvas scheme on a simple sensor network (Fig. 1). The node A wants to initiate the transmission of a message m which must contain the identification of the initiating node A. It selects two nodes

$$B \in N_1(A), \quad C \in N_2(A)$$

on a valid communication path determined by a routing algorithm.
The node A computes authenticators $h(m, K_{AB})$, $h(m, K_{AC})$ and sends the message $m, h(m, K_{AB})$, C, $h(m, K_{AC})$ to the node B.

In the message routing we consider two cases when a node receives a message. In the first one, a node processes a message which was just initiated by the original sender. The second case is more general, a node receives a message which was not sent by the original sender and has to forward it towards the selected destination.

- *1-hop after initiation.* The node B receives a message $m_1, \ldots, m_4$ from A. It checks whether m_1 contains the identification of the node A and compares the authenticator $h(m_1, K_{AB})$ with received m_2. If the tests succeed, the node B accepts the message. Then, it nominates a next node in the route following the node $m_3 = C$, for example, the node $D \in N_2(B)$. Then, the node B computes authenticators $h(m_1, K_{BC})$, $h(m_1, K_{BD})$ and finally sends the routed message m_1, A, $m_4, h(m_1, K_{BC})$, D, $h(m_1, K_{BD})$ to the node C.
- *Normal routing.* The node C receives a message $m_1, \ldots, m_6$ from B. It verifies whether
  - the node $A = m_2$ is a two-hop neighbor, i.e., $A \in N_2(C)$, and $h(m_1, K_{AC}) = m_3$;
  - B is a one-hop neighbor, i.e., $B \in N_1(C)$, and $h(m_1, K_{BC}) = m_4$.

  If the tests succeed, the node B accepts the message. The next hop D in the route has been already chosen by B. However, the node C selects the next-next hop $E \in N_2(C)$ from the set of 2-hop neighbors on the valid path. Finally, it sends the routed message m_1, B, m_6, $h(m_1, K_{CD})$, E, $h(m_1, K_{CE})$ to the node D. Similarly, the node D continues in routing of the message.

3. A model of the attacker and the security fallacy of the Canvas scheme

In this section we discuss a model of the attacker for formal analysis of WSN protocols with respect to formal analysis of the Canvas scheme. In the standard Dolev-Yao model, the attacker has the *complete control* over the communication network—he can eavesdrop, block, modify, insert, and replay messages. The attacker is also able to manipulate observed messages, i.e., compose and decompose them. On the other hand, the attacker is limited by *perfect cryptography*—he does not try to exploit any weakness in the cryptographic algorithms, but he can use some algebraic properties of cryptographic primitives. Furthermore, the communication network is left without details in the Dolev-Yao model. However, we need a formal model which adequately expresses the network topology and communication channels in order to analyze WSN security protocols. Moreover, we need to define the abilities of the attacker with respect to captured nodes. We discuss extensions of the Dolev-Yao model regarding to formal analysis of the Canvas scheme. Then, we describe the security fallacy of the Canvas scheme in the discussed models. Finally, we propose a solution for correcting the Canvas scheme.

3.1. Extensions of the Dolev-Yao model

Division into phases. The lifetime of a sensor network consists of several phases including: *pre-deployment initialization* of nodes; *physical nodes deployment*; *neighbor discovery*; *neighbor authentication*; *key establishment*; *operational phase* of the sensor network. Each phase of a sensor network provides diverse goals, takes various time, and requires different security assumptions. We can analyze each phase separately. Moreover, in each phase we can modify the model of the attacker in order to exactly define his abilities with respect to the character of the phase. During the analysis—in a certain phase—we can also assume that goals of the previous phases were achieved. For instance, in the case of the Canvas protocol we focus on the analysis in the operational phase of the sensor network and assume that the sensor network is already deployed. Moreover, each node knows its local topology and has established shared keys with nodes from the sets of one-hop and two-hop neighbors.

The weak model of the attacker. H. Vogt proposed [10] the adversary model in which the attacker is able to capture a sensor device, gain control over it, and reprogram it. On the other hand, only nodes that were presented in the previous phases of the sensor network are allowed to participate in the network and the attacker cannot have access to the network with his own device. Moreover, captured devices can communicate only via established communication links of the sensor network. There is no channel between them and the adversary. This way, we call this model of the attacker *weak*. In our opinion, it is problematic to model a cooperative strategy of captured nodes formally. This way, we consider this model unsuitable for a formal analysis.

D. Gollmann showed an attack on the Canvas protocol in the paper [6] assuming the above mentioned weak model. Let the nodes A, C from Fig. 1 be compromised and have a predefined cheating strategy. They want to cheat the node E with a fake message m' in which some honest node is marked as the originator. The honest node D initiates sending of a message m, i.e., nominates next hops A, B in the route, computes corresponding authenticators $h(m, K_{AD}), h(m, K_{BD})$, and sends the routed message to A. The compromised node A nominates C as the next hop following B. It computes authenticators $h(m, K_{AB}), h(m', K_{AE})$ and sends them along with $h(m, K_{BD})$ in the routed message to B. The node B cannot distinguish between the received fake authenticator $h(m', K_{AE})$ and the correct value $h(m, K_{AC})$. The node B follows routing of the message m, selects the node E as the next-next hop, computes MACs $h(m, K_{BC}), h(m, K_{BE})$, and sends them along with the fake authenticator $h(m', K_{AE})$ in the routed message to the node C. The compromised node C follows the predefined malicious strategy and receives the authenticator $h(m', K_{AE})$ for the fake message m' from B. It selects the node F as the next-next hop, computes $h(m', K_{CE}), h(m', K_{CF})$, and sends them along with $h(m', K_{AE})$

in the routed message to E. The node E checks whether the node C is a one--hop neighbor, the node A a two-hop neighbor, and all received authenticators. Then, it accepts the fake message m' even though compromised nodes A, C are isolated.

The strong model of the attacker. We propose a model of the attacker, which differs from the above mentioned model. The adversary is able to capture a device and copy all stored data including secrets from the device. Moreover, the attacker can use pre-established communication links of the captured devices by using his own device. Furthermore, we can define the initial knowledge of the attacker including the knowledge about the network topology and communication links. This way, we call this model *strong.* In our opinion, the strong model is more suitable for a formal analysis. Moreover, this model is "stronger" than the weak model, because we can easily simulate an attack from the weak model. Roughly speaking, the attacker knows all secrets from captured devices and a strategy of the captured nodes from the attack in the weak model. The attacker simulates the attack from the weak model following the predefined malicious strategy of the captured nodes. Note that the attack is realized by using an attacker's device which substitutes for the captured nodes in the sensor network.

In the strong model, the attack on the Canvas scheme is simpler and better explains the fallacy of the scheme. First, the attacker obtains all secrets from the captured nodes A, C including the keys K_{AE}, K_{CE}, K_{CF}. Then, he computes authenticators $h(m', K_{AE}), h(m', K_{CE}), h(m', K_{CF})$ and prepares the routed message of a fake message m' of the false route $A \to C \to E \to F$. Via the established channel between C and E he sends the routed message to the node E parading as C. The node E checks whether the node C is a one-hop neighbor, the node A is a two-hop neighbor, and all received authenticators. Then, the node E accepts the fake message m' and follows to propagate the message m' to the node F.

3.2. Correcting the Canvas protocol

In the above described security fault, we assume that captured nodes are isolated. A routed node checks two authenticators from previous hops in the route in order to verify the integrity of a message. One honest node from previous hops should protect against cheating by an adversary. However, there exists the above described attack. It is based on the fault of the protocol, which allows the node to accept the message in the route where two captured nodes directly follow. The node E accepts the message m' in the invalid route $A \to C \to E \to F$, even though the nodes A, C are captured.

In order to fix the scheme we need to avoid invalid routes in the sensor network as was suggested in [6]. However, it is sufficient to check whether previous hops in the route are on a valid communication path. This way, a node v during receiving a routed message from a previous hop p_1 and a previous-previous hop p_2 not only checks whether $p_1 \in N_1(v)$ and $p_2 \in N_2(v)$, but it needs to verify whether p_2p_1v is a valid path, i.e., $p_2p_1v \in \pi^2(G)$. Note that the knowledge of a node about the local topology was specified as an assumption in [11] and it is used implicitly in the routing algorithm by selecting a next-next hop on a valid path. Thus, we do not add new requirements for previous phases of the protocol. However, we specify explicitly what is necessary to assume and check in the operational phase of the Canvas protocol.

Example. In the example we show the proposed fix of the Canvas scheme on the simple sensor network from Fig. 1. By controlling the validity of the routed path of previous hops, the node E accepts a routed message from the previous hop C only when a previous-previous hop is B, or D. In the attack in the strong model, the attacker sends the message m', A, $h(m', K_{AE})$, F, $h(m', K_{CE})$, $h(m', K_{CF})$ to the node E parading as C. The node E—instead of checking whether C is a one-hop and A a two-hop neighbor—checks whether the path ACE is valid. The test fails and the node E correctly rejects the routed message without observing authenticators.

4. A formal model of the fixed Canvas protocol

4.1. The applied-pi calculus

The *applied pi-calculus* [1] is a language for describing concurrent processes and their interactions. It is based on the *pi-calculus*, but is intended to be less pure and therefore more convenient to use. Moreover, the applied pi-calculus allows us to define less usual communication and cryptographic primitives by defining function symbols and some equivalences on terms. We briefly describe the *syntax* and the *operational semantics* of the applied pi-calculus from the paper [4]. In the set of function symbols we distinguish between *constructors* and *destructors.* Constructors are used to build terms. On the other hand, destructors do not appear in terms, but only manipulate terms in processes. Moreover, we distinguish between *private* and *public* function symbols. Public function symbols can be used by an adversary, but he is forbidden to use private ones. The set of terms is built from *names*, *variables* and constructors applied to other terms. Note that the terms are *untyped* and the calculus is *monadic.*

Syntax and informal semantics. *Plain processes* are defined as follows. The *nil process* 0 does nothing; $\nu a.P$ generates a fresh name a and then behaves as P; **if** $M = N$ **then** P **else** Q behaves as P if $M = N$ and as Q otherwise; $P|Q$ executes P and Q in parallel; $!P$ generates an unbounded number of copies of P; **event** $M.P$ executes an event M and then behaves as P; $N(x).P$ receives a message M on a channel N and then behaves as $P\{M/x\}$, where x is bound to the input message M; $\overline{N}\langle M\rangle.P$ outputs a message M on a channel N and then executes P. Note that channels can be arbitrary terms, not only names. We abbreviate **if** $M = N$ **then** P when $Q = 0$ and a batch of parallel compositions $P_1|\ldots|P_n$ as $\prod_{i=1}^{n} P_i$.

The name a is bound in the process $\nu a.P$. The variable x is *bound* in P in the processes $N(x).P$ and **let** $x = M$ **in** P. A process is *closed* if it has no free variables; it may have free names. The process **let** $x = g(M_1,\ldots,M_n)$ **in** P **else** Q tries to evaluate $g(M_1,\ldots,M_n)$; if it succeeds, then x is bound to the result and P is executed, else Q is executed. More precisely, the semantics of a destructor g of arity n is given by a set $\mathrm{def}(g)$ of *rewrite rules* of the form $g(M_1,\ldots,M_n) \longrightarrow M$, where $M_1,\ldots,M_n,M$ are terms without names, and the variables of M also occur in $M_1,\ldots,M_n$. We extend these rules by $g(M'_1,\ldots,M'_n) \longrightarrow M'$ if and only if there exist a substitution σ and a rewrite rule $g(M_1,\ldots,M_n) \longrightarrow M$ in $\mathrm{def}(g)$ such that $M'_i = \sigma M_i$ for all $i \in \{1,\ldots n\}$, and $M' = \sigma M$.

Operational semantics and traces. We use the definition of *operational semantics* of the applied pi-calculus from the paper [4]. This semantics is superficially different from [1] where is defined using structural congruence relation and reduction relation on processes. However, it provides simplification in a definition of the data integrity property and in a proof that the property holds.

A *semantic configuration* is a pair $E,\mathcal{P}$ where the *environment* E is a finite set of names and $\mathcal{P}$ is a finite multiset of closed processes. The environment E must contain at least all free names of processes in $\mathcal{P}$. The semantic configuration $\{a_1,\ldots,a_n\},\{P_1,\ldots,P_m\}$ corresponds intuitively to the process $\nu a_1\ldots\nu a_n(P_1|\ldots|P_m)$. The semantics of the calculus is defined by a *reduction relation* $\longrightarrow$ on semantic configurations as shown in Fig. 2. Note that the equality in conditional rules 6–7 means syntactic equality.

We say that a *trace* $\mathcal{T} = E_0,\mathcal{P}_0 \longrightarrow^n E_n,\mathcal{P}_n$ *satisfies message* (N,M) and denote $\mathcal{T} \vDash^{\mathrm{m}} (N,M)$ if and only if the trace $\mathcal{T}$ contains a reduction $E,\mathcal{P} \cup \big\{\overline{N}\langle M\rangle.Q, N(x).P\big\} \longrightarrow E,\mathcal{P} \cup \big\{Q, P\{M/x\}\big\}$ for some $E,\mathcal{P},x,P,Q$. Intuitively, the trace satisfies message (N,M) when the message M has been sent on the channel N.

We say that a trace $\mathcal{T} = E_0,\mathcal{P}_0 \longrightarrow^n E_n,\mathcal{P}_n$ *satisfies event* M and denote $\mathcal{T} \vDash^{\mathrm{e}} M$ if and only if $\mathcal{T}$ contains a reduction $E,\mathcal{P} \cup \{\mathit{event}\ M.P\} \longrightarrow E,\mathcal{P} \cup \{P\}$ for some $E,\mathcal{P},P$. Intuitively, the trace satisfies event M when the event M has been executed.

We say that a trace $\mathcal{T}' = E'_0,\mathcal{P}'_0 \longrightarrow^q E'_q,\mathcal{P}'_q$ is a *subtrace* of a trace $\mathcal{T} = E_0,\mathcal{P}_0 \longrightarrow^n E_n,\mathcal{P}_n$ and denote $T' \preceq T$ if and only if $q \leq n$ and the trace $\mathcal{T}'$ is equal to the trace $\mathcal{T}$ up to qth reduction, i.e., $E'_0 = E_0, \mathcal{P}'_0 = \mathcal{P}_0, \ldots, E'_q = E_q, \mathcal{P}'_q = \mathcal{P}_q$.

1. $E, \mathcal{P} \cup \{0\} \longrightarrow E, \mathcal{P}$
2. $E, \mathcal{P} \cup \{!P\} \longrightarrow E, \mathcal{P} \cup \{P, !P\}$
3. $E, \mathcal{P} \cup \{P|Q\} \longrightarrow E, \mathcal{P} \cup \{P, Q\}$
4. $E, \mathcal{P} \cup \{\nu a.P\} \longrightarrow E \cup \{a'\}, \mathcal{P} \cup \big\{P\{a'/a\}\big\}$, where $a' \notin E$
5. $E, \mathcal{P} \cup \{\overline{N}\langle M\rangle.Q, N(x).P\} \longrightarrow E, \mathcal{P} \cup \big\{Q, P\{M/x\}\big\}$
6. $E, \mathcal{P} \cup \{\textbf{if } M = M \textbf{ then } P \textbf{ else } Q\} \longrightarrow E, \mathcal{P} \cup \{P\}$
7. $E, \mathcal{P} \cup \{\textbf{if } M = N \textbf{ then } P \textbf{ else } Q\} \longrightarrow E, \mathcal{P} \cup \{Q\}$, if $M \neq N$
8. $E, \mathcal{P} \cup \{\textbf{event } M.P\} \longrightarrow E, \mathcal{P} \cup \{P\}$
9. $E, \mathcal{P} \cup \{\textbf{let } x = g(M_1, \ldots, M_n) \textbf{ in } P \textbf{ else } Q\} \longrightarrow E, \mathcal{P} \cup \big\{P\{M'/x\}\big\}$
   if $g(M_1, \ldots, M_n) \longrightarrow M'$
10. $E, \mathcal{P} \cup \{\textbf{let } x = g(M_1, \ldots, M_n) \textbf{ in } P \textbf{ else } Q\} \longrightarrow E, \mathcal{P} \cup \{Q\}$
    if there exists no M' such that $g(M_1, \ldots, M_n) \longrightarrow M'$

FIGURE 2. Operational semantics of the applied pi-calculus.

4.2. The formal model of the fixed Canvas scheme

Public function symbols. For the sake of clarity, we use formatted routed messages R_1, R_2 with fixed number of fields in the formal model of the scheme. We introduce public constructors $R_1/5, R_2/7$. Public constructors $T_2/2, T_3/3$ are used for creating tuples. However, we denote tuples using standard notation as $(\_,\_)$, or $(\_,\_,\_)$. In addition, we introduce inverse, unary function symbols–
–public destructors $F_1^{R_1}, \ldots, F_5^{R_1}$ for selecting a field in the routed message R_1. This way, we add rewrite rules $F_i^{R_1}\big(R_1(x_1, \ldots, x_i, \ldots, x_5)\big) \longrightarrow x_i$. Similarly, we add rewrite rules for the public destructors $F_1^{R_2}, \ldots, F_7^{R_2}$ for the formatted message R_2 and public destructors $F_1^{T_2}, F_2^{T_2}, F_1^{T_3}, F_2^{T_3}, F_3^{T_3}$ in order to select a field in a tuple. Note that we use formatted messages only for clarity. They have similar properties as tuples and the attacker can modify the tags R_1, R_2. Therefore, the tags do not provide any security protection.

In order to model Message Authentication Codes we introduce a binary function symbol—a public constructor $h/2$, with no corresponding destructor. The fact that $h(m,k) = h(m',k')$ only when $m = m'$ and $k = k'$ models that h is *collision-free*. The absence of a destructor for h models the *one-wayness* of h.

Processes of the fixed Canvas protocol. We define a formal model of the Canvas scheme for a sensor network. The network is represented by a *communication graph* $G = (V, E)$, where $|V| = n_G$. We distinguish between two kinds of nodes: captured and honest. Let a set of captured nodes be $C \subseteq V$ and a set of honest nodes $V \setminus C$. Note that the communication graph and the set of captured nodes are *static* and do not change during the operational phase of the protocol.

An active adversary is represented by any closed process interacting with the below defined process $\mathsf{WSN}(G, C)$, which has a set of public names in its initial knowledge, can use public function symbols, and does not contain events. Actions of the attacker are not explicitly defined. However, the initial knowledge of the attacker contains a free name of the public "attacker" channel c_{A}. In the beginning, the attacker obtains identifications and channels of captured nodes from the process $\mathsf{WSN}(G, C)$. The attacker can use these channels for receiving messages. Moreover, he retrieves identifications and communication channels of nodes which are direct neighbors of captured nodes. The attacker can communicate via these channels. He also gains information about the local topology of the captured nodes and all shared keys of the captured nodes established with their neighbors up to the distance two. On the other hand, the attacker is forbidden to communicate via private channels.

The whole process of the sensor network $\mathsf{WSN}(G, C)$ is defined for a communication graph $G = (V, E)$ and a set of captured nodes $C \subseteq V$ as

$$\mathsf{WSN}(G, C) = \nu id_{v_1} \ldots \nu id_{v_{n_G}} \;.\; \nu c_{v_1} \ldots \nu c_{v_{n_G}} \;.\; \nu k_1 \ldots \nu k_t \;. \\ (\mathsf{Topology} \mid \mathsf{Select} \mid \mathsf{Select}^{\star} \mid \mathsf{Channel} \mid \mathsf{Key} \mid \textstyle\prod_{i=1}^{n_G} \mathsf{Node}).$$

First, for all nodes $v_1, \ldots, v_{n_G} \in V$ there are generated names $id_{v_1}, \ldots, id_{v_{n_G}}$ for their identification and fresh names $c_{v_1}, \ldots, c_{v_{n_G}}$ of their communication channels. There are also created all fresh keys $k_1, \ldots, k_t$ individually shared in each pair of nodes from V when their distance is up to two. Then, the whole process consists of the parallel composition of processes:

- $\mathsf{Topology}$ distributes mainly information about the local topology of nodes. First, the process executes an event $\mathsf{CAPTURED}(id_u)$ for each captured node $u \in C$ and an event $\mathsf{HONEST}(id_v)$ for each honest node $v \in V \setminus C$. Then, the process sends:
  - a message (id_v, c_v) on the "attacker" public channel c_{A} for each captured node $v \in C$ and for each $v \in V \setminus C$, such that there exist $u \in C$ and $u \in N_1(v)$;
  - a message $(id_{v_0}, id_{v_1}, id_{v_2})$ on the "attacker" public channel c_{A} for all nodes $v_0, v_1, v_2 \in V$, such that $v_0 \in C$ and $v_0 v_1 v_2 \in \pi^2(G)$;
  - a message (id_u, id_v, k_i) on the "attacker" public channel c_{A} for each key k_i shared between a captured node $u \in C$ and a node $v \in V$;
  - a message (id_v, c_v) on the private channel p^{Top} for each node $v \in V$.

  Later, the process provides information about the local topology of a node. For all nodes $v_0, v_1, v_2 \in V$ the process contains the parallel composition of $!\overline{p^{\mathrm{V}}(id_{v_2}, id_{v_1}, id_{v_0})}\langle \mathit{true} \rangle$, if $v_2 v_1 v_0 \in \pi^2(G)$ and $!\overline{p^{\mathrm{V}}(id_{v_2}, id_{v_1}, id_{v_0})}\langle \mathit{false} \rangle$, otherwise. This way, a node x_0 can check the validity of a path $x_2x_1x_0$ via the private channel $p^{\mathrm{V}}(x_2, x_1, x_0)$.

- Select chooses two nodes on a valid path starting in a requested node non--deterministically. For all nodes $v_0, v_1, v_2 \in V$, such that $v_0v_1v_2 \in \pi^2(G)$, the process contains the parallel composition of $!\overline{p^{\mathrm{S}}(id_{v_0})}\langle(id_{v_1}, id_{v_2})\rangle$. This way, a node x can receive a non-deterministic choice of two nodes on a valid path starting in x via the private channel $p^{\mathrm{S}}(x)$.
- Select* chooses a next-next hop on a valid path non-deterministically. For all nodes $v_0, v_1, v_2 \in V$, such that $v_0v_1v_2 \in \pi^2(G)$, the process contains the parallel composition of $!\overline{p^{\mathrm{S}^\star}(id_{v_0}, id_{v_1})}\langle id_{v_2}\rangle$. A node x can receive an nondeterministic choice of a next-next hop which follows a next hop y on a valid path via the private channel $p^{\mathrm{S}^\star}(x, y)$.
- Key provides access to established shared keys. In the beginning of the process WSN(G, C), exactly one key is generated for each pair of nodes when the distance between them is up to two. For each key k_i shared between nodes $u, v \in V$, such that the distance between them is up to two, the process contains the parallel composition of $!\overline{p^{\mathrm{K}}(id_u, id_v)}\langle k_i\rangle | !\overline{p^{\mathrm{K}}(id_v, id_u)}\langle k_i\rangle$. A node x can receive the key shared with a node y via the private channel $p^{\mathrm{K}}(x, y)$.
- Channel provides a name of the communication channel of a node for its direct neighbor. The process is defined as $\prod_{i=1}^{n_G} !\overline{p^{\mathrm{Ch}}(id_{v_i})}\langle c_{v_i}\rangle$. A node can receive the name of communication channel of a node x via the private channel $p^{\mathrm{Ch}}(x)$.
- Node describes the behavior of a sensor device as shown in Fig. 3.

First, the process Node obtains its identification id and the fresh channel c for listening and receiving messages via the private channel p^{Top} (line 1). Then, the process consists of the parallel composition of processes:

- Message initiation (lines 2–4). The process creates a new message m and marks the initiation of the message in the event INIT(m, id) (line 2). Via private channels, it receives nondeterministic selections of next hops n_1, n_2 on the valid path, corresponding keys k_1, k_2 individually shared with the selected next hops n_1, n_2, and the communication channel c_1 of the next hop n_1 (line 3). Finally, the process sends the routed message R_1 on the channel c_1 (line 4).
- 1-hop after initiation (lines 5–9). The process receives a routed message R_1 on its channel c from a sender p_1—the previous hop in the route (line 5). A next hop in the route is assigned in the variable n_1 and an authenticator designated for n_1 in the variable h_2 (line 5). Then, the process checks

```
    $Node \triangleq$
 1. $p^{\mathrm{Top}}(id, c)$ .
    ( //Message initiation
 2. $!\nu m$ . event $\mathsf{INIT}(m, id)$ .
 3. $p^{\mathrm{S}}(id)((n_1, n_2))$ . $p^{\mathrm{K}}(id, n_1)(k_1)$ . $p^{\mathrm{K}}(id, n_2)(k_2)$ . $p^{\mathrm{Ch}}(n_1)(c_1)$ .
 4. $\overline{c_1}\langle R_1(id, (id, m), h((id, m), k_1), n_2, h((id, m), k_2))\rangle$ ) |
    ( //1-hop after initiation
 5. $!c(R_1(p_1, (o, m), h_1, n_1, h_2))$ .
 6. if $o = p_1$ then $p^{\mathrm{K}}(id, p_1)(k_3)$ .
 7. if $h_1 = h((o, m), k_3)$ then event $\mathsf{ACCEPT}(m, o, id)$ .
 8. $p^{\mathrm{S}^\star}(id, n_1)(n_2)$ . $p^{\mathrm{K}}(id, n_1)(k_1)$ . $p^{\mathrm{K}}(id, n_2)(k_2)$ . $p^{\mathrm{Ch}}(n_1)(c_1)$ .
 9. $\overline{c_1}\langle R_2(id, (o, m), p_1, h_2, h((o, m), k_1), n_2, h((o, m), k_2))\rangle$ ) |
    ( //Normal routing
10. $!c(R_2(p_1, (o, m), p_2, h_2, h_1, n_1, h_3))$ .
11. $p^{\mathrm{V}}(p_2, p_1, id)(x)$ . if $x = true$ then
12. $p^{\mathrm{K}}(id, p_1)(k_1)$ . $p^{\mathrm{K}}(id, p_2)(k_2)$ .
13. if $h_1 = h((o, m), k_1)$ then if $h_2 = h((o, m), k_2)$ then
14. event $\mathsf{ACCEPT}(m, o, id)$ .
15. $p^{\mathrm{S}^\star}(id, n_1)(n_2)$ . $p^{\mathrm{K}}(id, n_1)(k_3)$ . $p^{\mathrm{K}}(id, n_2)(k_4)$ . $p^{\mathrm{Ch}}(n_1)(c_1)$ .
16. $\overline{c_1}\langle R_2(id, (o, m), p_1, h_3, h((o, m), k_3), n_2, h((o, m), k_4))\rangle$ )
```

FIGURE 3. The process Node.

whether the identification of the sender p_1 is correctly included in the received message as the originator of the message, i.e., $o = p_1$ (line 6). Then, it obtains a key k_3 shared with the previous hop p_1 and checks the validity of a received authenticator h_1 from the previous hop p_1 (lines 6–7). If the test succeeds, it marks the acceptance of the message m originated from the node $o = p_1$ in the event $\mathsf{ACCEPT}(m, o, id)$ (line 7). Via private channels, the process obtains a next-next hop n_2 in the route, corresponding keys k_1, k_2 individually shared with the selected next hops n_1, n_2, and the communication channel c_1 of the next hop n_1 (line 8). Finally, the process sends the routed message R_2 on the channel c_1 (line 9).

- Normal routing (lines 10–16). The process receives a message R_2 on its channel c from a sender p_1 – the previous hop in the route (line 10).

The received message R_2 contains an original message (o, m) from the originator o. A next hop is assigned in the variable n_1 and an authenticator designated for n_1 in the variable h_1. A previous-previous hop is assigned in the variable p_2 and an authenticator from the node p_2 in the variable h_2 (line 10). Then, the process checks the validity of the path p_2p_1id following the proposed fix of the scheme mentioned in Section 3 (line 11). If the test succeeds, the node obtains secret keys k_1, k_2 individually shared with the previous hops p_1, p_2 (line 12). The process also checks the validity of authenticators h_1, h_2 received from previous hops p_1, p_2 (line 13). If all the tests succeed, the process marks the acceptance of the message m originated from o in the event $\mathsf{ACCEPT}(m, o, id)$ (line 14). Via private channels, the process obtains a next-next hop n_2 in the route, corresponding keys k_3, k_4 individually shared with the selected next hops n_1, n_2, and the communication channel c_1 of the next hop n_1 (line 15). Finally, the process sends the routed message R_2 on the channel c_1 (line 16).

5. Formal analysis of the integrity property

Definition 1 (Data integrity). We say that a trace $\mathcal{T} = E_0, \mathcal{P}_0 \longrightarrow^n E_n, \mathcal{P}_n$ satisfies the data integrity property if and only if in all subtraces $\mathcal{T}' \preceq \mathcal{T}$ there holds:

$$\big(\mathcal{T}' \vDash^{\mathrm{e}} \mathsf{HONEST}(z) \wedge \mathcal{T}' \vDash^{\mathrm{e}} \mathsf{ACCEPT}(x, y, z)\big)$$
$$\Rightarrow \big(\mathcal{T}' \vDash^{\mathrm{e}} \mathsf{INIT}(x, y) \vee \mathcal{T}' \vDash^{\mathrm{e}} \mathsf{CAPTURED}(y)\big).$$

Intuitively, a trace satisfies the data integrity property when during all reductions of the trace if an honest node z accepts a message x originated from a node y, then the node y has already initiated the message x or the node y is captured.

Lemma 1. *For an arbitrary communication graph G, a set of captured nodes C, and for all traces $\mathcal{T}$ of the corresponding process $\mathsf{WSN}(G, C)$, in the presence of an active adversary,*

- $\mathcal{T} \vDash^{\mathrm{m}} (c, m')$ *and* $\mathcal{T} \vDash^{\mathrm{m}} \big(p^{\mathrm{K}}(r, s), k\big)$,
- $\big(F_4^{R_2}(m') = h((o, m), k)$, *or* $F_5^{R_2}(m') = h((o, m), k)$, *or* $F_3^{R_1}(m') = h((o, m), k)\big)$,
- $\mathcal{T} \vDash^{\mathrm{e}} \mathsf{HONEST}(r)$ *and* $\mathcal{T} \vDash^{\mathrm{e}} \mathsf{HONEST}(s)$, *imply that*

$\mathcal{T} \vDash^{\mathrm{e}} \mathsf{INIT}(m, o)$, *or* $\mathcal{T} \vDash^{\mathrm{e}} \mathsf{ACCEPT}(m, o, s)$, *or* $\mathcal{T} \vDash^{\mathrm{e}} \mathsf{ACCEPT}(m, o, r)$.

P r o o f. A routed message m' of the form R_1, or R_2 was sent on some channel c. The message contains a term $h\big((o,m),k\big)$. The key k is shared between two honest nodes r,s. With respect to the definition of the process $\mathsf{WSN}(G,C)$, the key k is not published on the "attacker" channel c_{A} in the process $\mathsf{Topology}$. According to the definition of the process Node, the creating and sending of the term $h\big((o,m),k\big)$ (Fig. 3 lines 4, 9, 16) by nodes r,s can be done after marking corresponding events. The term $h\big((o,m),k\big)$ could be created by the honest nodes r,s:

- in the message initiation (Fig. 3 line 4). Then ($r=o$, or $s=o$) and the event $\mathsf{INIT}(m,o)$ must be executed,
- in the message routing (Fig. 3 lines 9, 16). Then, at least one from the events $\mathsf{ACCEPT}(m,o,s)$, $\mathsf{ACCEPT}(m,o,r)$ must be executed.

Let us assume, that none from the events $\mathsf{ACCEPT}(m,o,r)$, $\mathsf{ACCEPT}(m,o,s)$, $\mathsf{INIT}(m,o)$ occurs in the trace $\mathcal{T}$. Therefore, the term $h\big((o,m),k\big)$ must be created by the attacker. For this purpose, the key k must be in his knowledge. On the other hand, the key k is not published on the attacker channel c_{A} in the process $\mathsf{Topology}$, since the nodes r,s are honest. Moreover, the attacker is forbidden to use private channels $p^{\mathrm{K}}(r,s)$, $p^{\mathrm{K}}(s,r)$. He is not able to deduce k from eavesdropping the communication in the trace $\mathcal{T}$, because the key k can be sent by honest nodes r,s (Fig. 3 lines 4, 9, 16) only in a term $h(\_,k)$ and in the set of function symbols there is no destructor for the function $h/2$. Therefore, at least one event from $\mathsf{INIT}(m,o)$, $\mathsf{ACCEPT}(m,o,s)$, $\mathsf{ACCEPT}(m,o,r)$ must occur in the trace $\mathcal{T}$. □

Theorem 2. *For an arbitrary communication graph G and a set of captured nodes C, assuming that captured nodes are isolated in G, it holds that all traces $\mathcal{T}$ of the corresponding process $\mathsf{WSN}(G,C)$, in the presence of an active adversary, satisfy the data integrity property.*

P r o o f. We prove the theorem by contradiction. Let us assume that the statement is not true. Therefore, there exists some communication graph G' and the set of captured nodes C', where captured nodes from C' are isolated in G'. There also exists a closed process that represents the active adversary interacting with the process $\mathsf{WSN}(G',C')$. For the whole process there exists a trace which does not satisfy the integrity property. Hence, there must exist its subtrace $\mathcal{T}' = E'_0,\mathcal{P}'_0 \longrightarrow^n E'_n,\mathcal{P}'_n$, such that m',o',id' exist and $\mathcal{T}' \vDash^{\mathrm{e}} \mathsf{ACCEPT}(m',o',id')$, $\mathcal{T}' \vDash^{\mathrm{e}} \mathsf{HONEST}(id')$, $\mathcal{T}' \nvDash^{\mathrm{e}} \mathsf{INIT}(m',o')$, $\mathcal{T}' \nvDash^{\mathrm{e}} \mathsf{CAPTURED}(o')$.

From the trace $\mathcal{T}'$, we take two subtraces $\mathcal{T}'',\mathcal{T}''' \preceq \mathcal{T}'$, in order to have two traces $\mathcal{T}'' = E'_0,\mathcal{P}'_0 \longrightarrow^q E'_q,\mathcal{P}'_q$ and $\mathcal{T}''' = E'_0,\mathcal{P}'_0 \longrightarrow^{q-1} E'_{q-1},\mathcal{P}'_{q-1}$, where $q \le n$. Moreover, there exists $id'' \in E'_{q-1}$ such that $\mathcal{T}'' \vDash^{\mathrm{e}} \mathsf{ACCEPT}(m',o',id'')$, $\mathcal{T}''' \vDash^{\mathrm{e}} \mathsf{HONEST}(id'')$, and there is no $x \in E'_{q-1}$ such that $\mathcal{T}''' \vDash^{\mathrm{e}} \mathsf{HONEST}(x)$

and $\mathcal{T}''' \models^{\mathrm{e}} \mathsf{ACCEPT}(m', o', x)$. From the trace $\mathcal{T}'$, it holds $\mathcal{T}''' \not\models^{\mathrm{e}} \mathsf{INIT}(m', o')$ and $\mathcal{T}''' \not\models^{\mathrm{e}} \mathsf{CAPTURED}(o')$. Intuitively, the honest node id'' is the first honest node that accepts the message m' originated from o' exactly at qth reduction step in the trace $\mathcal{T}'$. Therefore, in the trace $\mathcal{T}'''$, which is a subtrace of $\mathcal{T}'$ of the length $q-1$ reductions, the message m' originated from o' is not accepted by any honest node x. Note that the node id'' can be the same node as id'.

According to the definition of the process Node, the node id'' receives its name id'' and the channel name c'' on the private channel p^{Top}. More formally, there exists $c'' \in E'_{q-1}$, such that $\mathcal{T}''' \models^{\mathrm{m}} \big(p^{\mathrm{Top}}, (id'', c'')\big)$.

We consider two cases, when the event $\mathsf{ACCEPT}(m', o', id'')$ could occur in the trace $\mathcal{T}''$:

- 1-hop after initiation. According to the definition of the appropriate part of the process Node (Fig. 3 lines 5–7), there must exist $k'' \in E'_{q-1}$, such that

$$\mathcal{T}''' \models^{\mathrm{m}} \big(p^{\mathrm{K}}(id'', o'), k''\big),\ \mathcal{T}''' \models^{\mathrm{m}} \big(c'', R_1(o', (o', m'), h((o', m'), k''), \_, \_)\big).$$

  With respect to the definition of the process $\mathsf{Topology}$ from the fact $\mathcal{T}''' \not\models^{\mathrm{e}} \mathsf{CAPTURED}(o')$ we obtain $\mathcal{T}''' \models^{\mathrm{e}} \mathsf{HONEST}(o')$. However, together with $\mathcal{T}''' \not\models^{\mathrm{e}} \mathsf{ACCEPT}(m', o', o')$, $\mathcal{T}''' \not\models^{\mathrm{e}} \mathsf{ACCEPT}(m', o', id'')$, and $\mathcal{T}''' \not\models^{\mathrm{e}} \mathsf{INIT}(m', o')$, this contradicts Lemma 1.

- Normal routing. According to the definition of the appropriate part of the process Node (Fig. 3 lines 10–14), there must exist $k''_1, k''_2, p''_1, p''_2 \in E'_{q-1}$, such that

$$\mathcal{T}''' \models^{\mathrm{m}} (p^{\mathrm{K}}(id'', p''_1), k''_1),\ \mathcal{T}''' \models^{\mathrm{m}} \big(p^{\mathrm{K}}(id'', p''_2), k''_2\big),$$

  and $\mathcal{T}''' \models^{\mathrm{m}} \big(c'', R_2(p''_1, (o', m'), p''_2, h\big((o', m'), k''_2\big), h((o', m'), k''_1), \_, \_)\big)$. We observe all possibilities according to the honesty of the previous hops p''_1, p''_2:
  - Both previous hops are captured, i.e., $\mathcal{T}''' \models^{\mathrm{e}} \mathsf{CAPTURED}(p'')$ and $\mathcal{T}''' \models^{\mathrm{e}} \mathsf{CAPTURED}(p''_2)$. With respect to the definition of the process Node (Fig. 3 line 11), we have $\mathcal{T}''' \models^{\mathrm{m}} \big(p^{\mathrm{V}}(p''_2, p''_1, id''), \mathit{true}\big)$. According to the definition of the process $\mathsf{Topology}$, the path $p''_2 p''_1 id''$ is the valid path of the length two. Nevertheless, the existence of the captured nodes p''_1, p''_2 which are direct neighbors contradicts the assumption.
  - At least one from previous hops is honest, i.e., $\mathcal{T}''' \models^{\mathrm{e}} \mathsf{HONEST}(p'')$, where $p'' \in \{p''_1, p''_2\}$. Nevertheless, facts $\mathcal{T}''' \not\models^{\mathrm{e}} \mathsf{ACCEPT}(m', o', id'')$, $\mathcal{T}''' \not\models^{\mathrm{e}} \mathsf{ACCEPT}(m', o', p'')$, and $\mathcal{T}''' \not\models^{\mathrm{e}} \mathsf{INIT}(m', o')$ are in contradiction with Lemma 1.

□

6. Conclusion

The Dolev-Yao model does not adequately express the network topology and communication channels of security protocols for WSNs. Developing a more detailed model of the attacker was formulated as a challenge in the computer security research field [6]. In the paper we built a formal model of the Canvas scheme in the applied pi-calculus. This model includes a model of the network topology, communication channels, captured nodes, and capabilities of the attacker. Formal modeling of the scheme helped us to better understand the fallacy of the scheme. We proposed a solution for correcting the Canvas scheme, which is based on explicit specification of an assumption and adding some check in the protocol.

In the semantic model of the applied pi-calculus we specified the data integrity property of the scheme. We proved that the fixed Canvas scheme in the presence of an active adversary ensures the data integrity of messages. We proved this property for an arbitrary communication graph and a set of captured nodes, assuming that captured nodes are isolated in the communication graph.

The proposed formal model could be applied to other security protocols as well. The applied pi-calculus allows us to model various cryptographic primitives. Although we analyzed the Canvas scheme only manually, the ProVerif tool [5] can be used for the automatical verification of correspondences [4]. However, an automatical analysis could be done for a concrete communication graph and a set of captured nodes. On the other hand, they can be generated automatically, for example, by exhaustive listening of each other isomorphic communication graphs with the fixed number of nodes.

REFERENCES

[1] ABADI, M. — FOURNET, C.: *Mobile values, new names, and secure communication*, in: Proc. of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, ACM SIGPLAN Notices, Vol. 36, ACM, New York, 2001, pp. 104–115.

[2] ABADI, M. — GORDON, A. D.: *A calculus for cryptographic protocols: the spi calculus*, in: Proc. of the 4th ACM Conf. on Computer and Communications Security—CCS '97, ACM, New York, 1997, pp. 36–47.

[3] ARMANDO, A. ET AL.: *The AVISPA tool for the automated validation of internet security protocols and applications*, in: Proc. of the 17th Internat. Conf. on Computer Aided Verification—CAV '05 (K. Etessami et al., eds.), Lect. Notes Comput. Sci., Vol. 3576, Springer, Berlin, 2005, pp. 281–285.

[4] BLANCHET, B.: *Automatic verification of correspondences for security protocols*, J. Comput. Secur. **17** (2009), 363–434.

[5] BLANCHET, B.: *An efficient cryptographic protocol verifier based on prolog rules*, in: Proc. of 14th IEEE Computer Security Foundations Workshop—CSFW '01, IEEE Comput. Soc., Washington, 2001, pp. 82–96.

[6] GOLLMANN, D.: *Protocol analysis for concrete environments*, in: Proc. of the 10th Internat. Conf. on Computer Aided Systems Theory—EUROCAST '05 (R. Moreno-Díaz et al., eds.), Lect. Notes Comput. Sci., Vol. 3643, Springer, Berlin, 2005, pp. 365–372.

[7] MENEZES, A. — VAN OORSHOT, P. — VANSTONE, S.: *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL, 1997.

[8] TOBARRA, L. ET AL.: *Model checking wireless sensor network security protocols: TinySec + LEAP*, in: Proc. of IFIP WG 6.8 1st Internat. Conf. on Wireless Sensor and Actor Networks—WSAN '07 (L. Orozco-Barbosa et al., eds.), IFIP Adv. Inform. Commun. Technol., Vol. 248, Springer, Berlin, 2008, pp. 95–106.

[9] VOGT, H.: *Exploring message authentication in sensor networks*, in: Proc. of the 1st European Workshop on Security in Ad-hoc and Sensor Networks—ESAS '04 (C. Castelluccia et al., eds.), Lect. Notes Comput. Sci., Vol. 3313, Springer, Berlin, 2005, pp. 19–30.

[10] VOGT, H.: *Increasing attack resiliency of wireless ad hoc and sensor networks*, in: Proc. of the 2nd Internat. Workshop on Security in Distributed Comput. Systems—ICDCSW '05, Vol. 2., IEEE Computer Society, Washington, 2005, pp. 179–184.

[11] VOGT, H.: *Integrity preservation for communication in sensor networks*, Tech. Report 434, Institute for Pervasive Computing, ETH Zürich, 2004.

[12] DU, X.—XIAO, Y.: *A survey on sensor network security,* in: Wireless Sensor Networks and Applications (Li, Y., et al., eds.), Signals and Communication Technology Series, New York, 2008, pp. 403–421.

Received September 30, 2009

Marián Novotný
Institute of Computer Science
Faculty of Science
Pavol Jozef Šafárik University
Jesenná 5
SK–040 01–Košice
SLOVAKIA
E-mail: majo.novotny@gmail.com