Design and Analysis of a Generalized Canvas Protocol*

Marián Novotný

Institute of Computer Science, Pavol Jozef Šafárik University,
Jesenná 5, 041 54 Košice, Slovakia
marian.novotny@upjs.sk

Abstract. The Canvas protocol was developed by Harald Vogt [10] and should provide data integrity in Wireless Sensor Networks. However, Dieter Gollmann published [5] an attack on the protocol. This example supports a widespread belief that design of security protocols is notoriously error-prone. Therefore, it is required to use formal methods to analyze their security properties. In the paper we present design and analysis of a generalized Canvas protocol. We consider the fallacy of the Canvas scheme in different models of the attacker and present a solution for correcting the scheme. We discuss a motivation for generalization of the Canvas protocol and introduce a k-generalized version of the scheme for some parameter $k \geq 2$. We build a formal model of the k-generalized 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. In the semantic model of the applied pi-calculus we specify the data integrity property of the scheme. We prove that the proposed k-generalized Canvas scheme, in the presence of an active adversary, provides data integrity of messages assuming that at least one honest node exists on each path of the length $k-1$ in the communication graph of a sensor network. Finally, we discuss the usability of the proposed formal model for 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 above mentioned applications of sensor networks, it is important to ensure security properties such as *confidentiality, authenticity*, and *data integrity*. Traditional security techniques as protocols for *key establishmen*t and *authentication* cannot be applied directly, because sensor devices are limited

* This work was partially supported by APVV grant SK-SI-0010-08 and Slovak VEGA grant 1/0035/09.

P. Samarati et al. (Eds.): WISTP 2010, LNCS 6033, pp. 106–121, 2010.

in their computation, energy, and communication capabilities. Moreover, they are often deployed in accessible areas, where *capture* by an attacker is possible. A sensor device is not considered as *tamper-resistant* in literature. 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 captured device and fully determine its operation. Furthermore, the typical communication in a sensor network is not *point-to-point*, but *one-to-many* or *many-to-one*. Therefore, design of WSN security protocols has become a challenge in the computer security research field. Many protocols originally designed for WSNs have been developed [11]. We focus on the Canvas scheme [10] 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 [7]. The authors used the *model checking* tool AVISPA [2] and modeled and analyzed TinySec [11] combined with LEAP [11] protocol in various communication scenarios. They analyzed protocols in the standard *Dolev-Yao model* of the attacker. In this model, the communication network is described undetailed. 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. Development of more detailed model of the attacker was formulated as a challenge in the computer security research field [5]. We propose a formal model of a generalized Canvas scheme in the applied pi-calculus, which consists of 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, discusses the fallacy of the scheme in different models of the attacker, and provides a solution for correcting the scheme. In Section 3 we discuss a motivation for generalization of the Canvas protocol and propose a generalized Canvas scheme. In Section 4 we present a formal model of the proposed scheme in the applied pi-calculus. In Section 5 we formulate and analyze the data integrity property of the proposed scheme in the semantic model of the applied pi-calculus. The last section presents our conclusions and suggestions for future work and discusses the usability of the proposed formal model for 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 a length l in a graph G. We define a *set of i-hop*

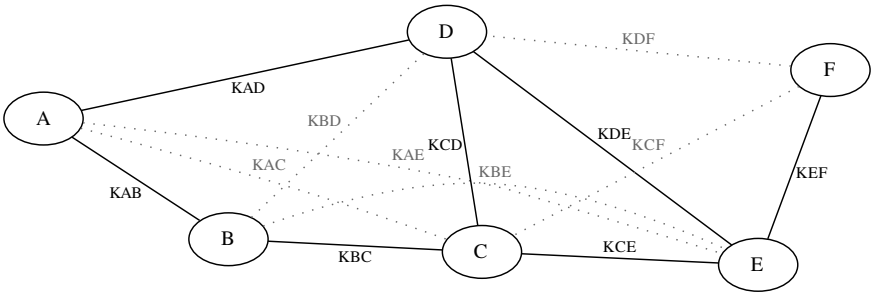

Fig. 1. A simple communication graph with established keys in the Canvas scheme

neighbors $N_i(v)$ of a node $v \in V$ as the set of nodes $u \in V$ such that exists a path $\rho \in \pi^i(G)$ starting in v and ending in u. The Canvas protocol was designed by H. Vogt and published in [8,9,10]. It should provide *data integrity* or *data origin authentication* [6] in a sensor network under assumption that captured nodes are *isolated*, i. e., captured nodes are not direct neighbors in a communication graph. Each node shares secret keys with nodes that are one or two hops away as shown in Fig. 1. *Message authentication codes – MACs* [6] are used to protect the integrity of a message between nodes which share the secret key. Let K_{AB} denote the key shared between nodes A and 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 explain 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 its identification. It selects two nodes $B \in N_1(A), C \in N_2(A)$ 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 message routing we consider two cases when a node receives a message. In the first one, the node processes the message which was just initiated by an original sender. 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.

In the general case of message routing – following the example – the node C receives a message $m_1, \ldots, m_6$ from B. It verifies whether $A = m_2$ is a two-hop neighbor, i. e., $A \in N_2(C)$, and $h(m_1, K_{AC}) = m_3$; the sender 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.

The Security Fallacy of the Scheme. D. Gollmann showed an attack on the Canvas protocol in the paper [5]. He assumes an attacker 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. 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.

Let the nodes A, C from Fig. 1 be compromised and have 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 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.

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 about the network topology and communication links. In this model, the attack on the Canvas scheme is simpler and explains the fallacy of the scheme better. On the other hand, the model is stronger (we can easily simulate an attack from the above mentioned model) and more suitable for a formal analysis. 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 parades 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.

In order to fix the scheme we need to avoid invalid routes in the sensor network as was suggested in [5]. 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 [10] 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.

3 Generalized Canvas Scheme

3.1 Motivation

Our aim is to generalize the Canvas scheme, in order to protect the data integrity of the message witnessed by k previous hops in the routing. We assume that each node v has established shared keys with nodes from the sets of neighbors $N_1(v), \ldots, N_k(v)$. Moreover, the node v knows information about its local topology up to the distance k. This way the node v can check the validity of a path of the length k ending in v.

The k-generalized Canvas scheme should provide data integrity under the assumption that at least one node which is not captured exists on each path of the length $k-1$ in the communication graph. We distinguish between two kinds of sensor devices – *protected* and *unprotected*. The attacker is unable to copy secrets from a protected device. This property can be realized by making the protected device tamper-resistant or placing the protected device on a safe location, where capture is problematic. On the other hand, an unprotected device can be captured by the attacker, who can also copy secrets from the device and gain control over it. During the deployment and initialization of a sensor network, it should be ensured, that at least one protected node exists on each path of the length $k-1$ in the communication graph. This can be realized by appropriate geometrical covering of an area of the sensor network by protected devices during the deployment. The initialization phase of the sensor network is realized with respect to the required property and the communication graph is modified when necessary. Another possibility is to randomly distribute protected and unprotected devices with sufficient ratio of protected ones and maintain a communication graph with respect to the required property. Finally, the last idea is based on a planned deployment of the sensor network with minimization of the number of protected devices due to increased costs. Building a topology of a sensor network with the minimal number of protected devices could be a problem of independent interest in the field of combinatorial optimization.

3.2 A Proposed Scheme

Let us denote a *concatenation of strings* s_1, s_2 as $s_1||s_2$ and a *string representation* of an integer $i \leq k$ as $'i'$. We define a *chain of message authentication codes* of a message m under keys $\langle K_1, \ldots, K_n \rangle$ as

$$H(m, \langle K_1, \ldots, K_n \rangle) = \begin{cases} h('1'||m, K_1), & \text{if } n = 1; \\ h('n'||m||H(m, \langle K_1, \ldots, K_{n-1} \rangle), K_n), & \text{otherwise.} \end{cases}$$

Message initiation. Let us assume that there exists a routing algorithm, which selects a routing path $\rho = A_0 \ldots A_n$. The node A_0 wants to initiate the transmission of a message $m = A_0||m'$ which contains its identification. It selects k nodes $A_1, \ldots, A_k$ on the valid communication path starting in A_0 by the routing algorithm. It computes an authenticator $h('1'||m, K_{A_0A_i})$ for each $i \leq k$ and sends the message $m, A_0, \ldots, A_k, h('1'||m, K_{A_0A_1}), \ldots, h('1'||m, K_{A_0A_k})$ to the node A_1.

Message routing. We consider k cases when a node receives a message. In the first $k-1$ cases, the node processes the message routed up to $k-1$ hops since the initiation. The last case is more general, the node receives the message which has been routed at least $k-1$ steps and has to forward it towards selected destinations.

- *1-hop after initiation.* The node A_1 receives a message $m, A_0, \ldots, A_k$, $h_1, \ldots, h_k$ from A_0. It checks whether m contains the identification of the node A_0 and the path A_0A_1 is valid, i. e., the node A_0 is its direct neighbor. Then, it compares the authenticator $h('1'||m, K_{A_1A_0})$ with h_1. If the tests succeed, the node A_1 accepts the message. It nominates a next node in the route following the node A_k, i. e., the node A_{k+1} in the selected path ρ. Then, it computes the authenticator $h'_{k+1} = h('1'||m, K_{A_1A_{k+1}})$ designated for the node A_{k+1}. It also updates received authenticators, i. e., for each $j \in \{2, \ldots, k\}$ the node computes $h'_j = H(m, \langle K_{A_0A_j}, K_{A_1A_j} \rangle) = h('2'||m||h_j, K_{A_1A_j})$. Finally, it sends the routed message $m, A_0, \ldots, A_{k+1}$, $h'_2, \ldots, h'_{k+1}$ to the node A_2.
- *i-hop after initiation.* In this paragraph we describe general scenario of routing of a message up to $k-1$ steps since the initiation that includes the first step mentioned above. The node A_i receives a message $m, A_0, \ldots, A_{k+i-1}, h_i, \ldots, h_{k+i-1}$ from the node A_{i-1}. It checks whether m correctly contains the identification of the node A_0 and the path $A_0 \ldots A_i$ is valid. Then, it compares the chain of MACs $H(m, \langle K_{A_0A_i}, \ldots, K_{A_{i-1}A_i} \rangle)$ with received h_i. If the tests succeed, it nominates the next node A_{k+i} in the route following the node A_{k+i-1} and computes the corresponding authenticator $h'_{k+i} = h('1'||m, K_{A_iA_{k+i}})$. It updates all received authenticators, i. e., for each $j \in \{i+1, \ldots, k\}$ it computes $h'_j = H(m, \langle K_{A_0A_j}, \ldots, K_{A_iA_j} \rangle) = h('i+1'||m||h_j, K_{A_iA_j})$ and if $i > 1$ for each $j \in \{k+1, \ldots, k+i-1\}$ computes $h'_j = H(m, \langle K_{A_{j-k}A_j}, \ldots, K_{A_iA_j} \rangle) = h('i-j+k+1'||m||h_j, K_{A_iA_j})$. Finally, it sends the message $m, A_0, \ldots, A_{k+i}, h'_{i+1}, \ldots, h'_{k+i}$ to the node A_j.

– *Normal routing* of a message at l-th step since the initiation, where $l > k-1$. The node A_l receives a message $m, A_{l-k}, \ldots, A_{l+k-1}, h_l, \ldots, h_{l+k-1}$ from the node A_{l-1}. It checks whether the path $A_{l-k} \ldots A_l$ is valid. Then, it computes the chain of authenticators $H(m, \langle K_{A_{l-k}A_l}, \ldots, K_{A_{l-1}A_l} \rangle)$ and compares it with received value h_l. If the tests succeed, it accepts the message. Then, it nominates the next node A_{l+k} following the node A_{l+k-1} in the route and computes the authenticator $h'_{l+k} = h('1'||m, K_{A_l A_{l+k}})$ designated for the node A_{l+k}. It also updates received authenticators, i. e., it computes $h'_j = H(m, \langle K_{A_{l-k+1}A_j}, \ldots, K_{A_l A_j} \rangle) = h('k+l+1-j'||m||h_j, K_{A_l A_j})$ for each $j \in \{l+1, \ldots, l+k-1\}$. Finally, it sends the message $m, A_{l-k+1}, \ldots, A_{l+k}$, $h'_{l+1}, \ldots, h'_{l+k}$ to the node A_{l+1}.

Note that during the update of an authenticator $H(m, \langle K_1, \ldots, K_j \rangle) = h('j'||m||$ $H(m, \langle K_1, \ldots, K_{j-1} \rangle), K_j)$ from received $H(m, \langle K_1, \ldots, K_{j-1} \rangle)$ a node cannot verify the validity of $H(m, \langle K_1, \ldots, K_{j-1} \rangle)$. However, the node needs to check the right type, i. e., the right size of the received authenticator in order to avoid a type-flaw attack.

For the sake of simplicity, we do not define the process of completing the routing. In the case, when a node does not want to continue in routing of a message, it does not select a next node following the last chosen and also skips the corresponding authenticator. However, the node has to forward the message with updated authenticators to already selected next node. Similarly, already chosen nodes – up to the last one – continue to complete the routing.

4 A Formal Model of the k-Generalized Canvas Scheme

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 [3]. 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

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. $E, \mathcal{P} \cup \{\nu a.P\} \longrightarrow E \cup \{a'\}, \mathcal{P} \cup \{P\{a'/a\}\}$, where $a' \notin E$
5. $E, \mathcal{P} \cup \{\overline{N}\langle M\rangle.Q, N(x).P\} \longrightarrow E, \mathcal{P} \cup \{Q, P\{M/x\}\}$
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. $E, \mathcal{P} \cup \{\textbf{let } x = g(M_1, \ldots, M_n) \textbf{ in } P \textbf{ else } Q\} \longrightarrow E, \mathcal{P} \cup \{P\{M'/x\}\}$ 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 $g(M_1, \ldots, M_n) \longrightarrow M'$

Fig. 2. Operational semantics

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 $Q = 0$, a batch of parallel compositions $P_1|\ldots|P_n$ as $\prod_{i=1}^{n} P_i$, and a sequence of executions $P_1 \ldots P_n$ as $\sum_{i=1}^{n} P_i$.

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 [3]. 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 $E, \mathcal{P} \cup \{\overline{N}\langle M\rangle.Q, N(x).P\} \longrightarrow E, \mathcal{P} \cup \{Q, P\{M/x\}\}$ 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 $E, \mathcal{P} \cup \{event\ M.P\} \longrightarrow E, \mathcal{P} \cup \{P\}$ 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 q-th reduction, i. e., $E'_0 = E_0, \mathcal{P}'_0 = \mathcal{P}_0, \ldots, E'_q = E_q, \mathcal{P}'_q = \mathcal{P}_q$.

4.2 A Formal Model of the Scheme

Public Function Symbols. For the sake of clarity in the formal model of the scheme, we use formatted routed messages $R_1, \ldots, R_k$ with fixed number of fields. We introduce public constructors $R_k/3k+1, R_i/2k+i+1$, for all $i < k$. Public constructors T_n/n, for all $2 \leq n \leq k+1$ are used for creating n-tuples. However, we denote tuples using standard notation as $(\_, \ldots, \_)$. In addition, we introduce inverse, unary function symbols – public destructors $F_1^{R_1}, \ldots, F_{2k+2}^{R_1}$ for selecting a field in the routed message R_1. This way, we add rewrite rules $F_i^{R_1}(R_1(x_1, \ldots, x_i, \ldots, x_{2k+2})) \longrightarrow x_i$. Similarly, we add rewrite rules for public destructors $F_i^{R_j}$ in order to select i-th field in a formatted message R_j and for public destructors $F_i^{T_n}$ in order to select i-th field in n-tuple. Note that we use formatted messages only for clarity. They have similar properties as tuples and the attacker can modify the tags $R_1, \ldots, R_k$. 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. We do not introduce a special function symbol for the chain of message authentication codes. We use the notation $H(m, \langle K_1, \ldots, K_n \rangle)$ as an abbreviation of the recursively defined term $H(m, \langle K_1, \ldots, K_n \rangle)$ as $h((\mathbf{1}, m), K_1)$, if $n = 1$, and $H(m, \langle K_1, \ldots, K_n \rangle) = h((\mathbf{n}, (m, H(m, \langle K_1, \ldots, K_{n-1} \rangle))), K_n)$, otherwise. For this purpose, we add public constant constructors $\mathbf{1}/0, \ldots, \mathbf{k}/0$ into the set of function symbols. Note that the concatenation of strings is represented by tuples. Both string representations of the depth of the chain of MACs and MACs have known size. This way, one can select a field of the concatenated message similar to tuples.

Lemma 1. *$H(m, \langle K_1, \ldots, K_n \rangle) = H(m', \langle K'_1, \ldots, K'_{n'} \rangle)$ if and only if $m = m'$, $n = n'$ and $K_i = K'_i$ for all $i \leq n$.*

Proof. The direction $\Leftarrow$ is trivial. We prove the opposite direction by induction according to the depth n of the term $H(m, \langle K_1, \ldots, K_n \rangle)$. If $n = 1$, then from the definition $H(m, \langle K_1 \rangle) = h((\mathbf{1}, m), K_1)$. If $n' > 1$ then $H(m', \langle K'_1, \ldots, K'_{n'} \rangle) =$

$h((\mathbf{n}', (m', H(m', \langle K'_1, \ldots, K'_{n-1'}\rangle))), K'_n) \neq h((\mathbf{1}, m), K_1)$. Therefore $n' = 1$ and $h((\mathbf{1}, m), K_1) = h((\mathbf{1}, m'), K'_1)$ only when $m = m'$ and $K_1 = K'_1$.

Let us assume that for $j < n$ is the statement true and $n > 1$. We have $H(m, \langle K_1, \ldots, K_n\rangle) = h((\mathbf{n}, (m, H(m, \langle K_1, \ldots, K_{n-1}\rangle))), K_n)$. If $n' = 1$, then $h((\mathbf{1}, m'), K'_1) \neq h((\mathbf{n}, (m, H(m, \langle K_1, \ldots, K_{n-1}\rangle))), K_n)$. Therefore it must hold that $n' > 1$ and from the equation $h((\mathbf{n}, (m, H(m, \langle K_1, \ldots, K_{n-1}\rangle))), K_n) = h((\mathbf{n}', (m', H(m', \langle K'_1, \ldots, K'_{n'-1}\rangle))), K'_{n'})$ we obtain two equalities $K_n = K'_{n'}$ and $(\mathbf{n}, (m, H(m, \langle K_1, \ldots, K_{n-1}\rangle))) = (\mathbf{n}', (m', H(m', \langle K'_1, \ldots, K'_{n'-1}\rangle)))$. Consequently, it holds $H(m, \langle K_1, \ldots, K_{n-1}\rangle) = H(m', \langle K'_1, \ldots, K'_{n'-1}\rangle)$, $m = m'$, and $n = n'$. Finally, from the induction hypothesis we obtain $m = m', n = n'$ and $K_i = K'_i$ for all $i \leq n$.

Processes of the k-Generalized Canvas protocol. We define a formal model of the k-generalized Canvas scheme for a sensor network and a fixed security parameter $k \geq 2$. 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 scheme.

An active adversary is represented by any closed process interacting with the below defined process $\mathsf{Network}(G, C, k)$, 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 $\mathsf{Network}(G, C, k)$. 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 all established keys of the captured nodes shared with their neighbors up to the distance k and information about the local topology of the captured nodes. On the other hand, the attacker is forbidden to communicate via private channels.

We define the whole process of the sensor network $\mathsf{Network}(G, C, k)$ for a security parameter k and a communication graph $G = (V, E)$ with respect to a set of captured nodes C as

$$\mathsf{Network}(G, C, k) = \nu id_{v_1} \ldots \nu id_{v_{n_G}} . \nu c_{v_1} \ldots \nu c_{v_{n_G}} . \nu K_1 \ldots \nu K_t . \\ (\mathsf{Topology} \mid \mathsf{Select} \mid \mathsf{Select}^\star \mid \mathsf{Channel} \mid \mathsf{Key} \mid \textstyle\prod_{i=1}^{n_G} \mathsf{Node}).$$

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 k. 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 exists $u \in C$ and $u \in N_1(v)$;
- a message $(id_{v_0}, \ldots, id_{v_k})$ on the "attacker" public channel c_A, for all nodes $v_0, \ldots, v_k \in V$, if $v_0 \in C$ and $v_0 \ldots v_k \in \pi^k(G)$;
- 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 each $l \in \{1, \ldots, k\}$ and all nodes $v_0, \ldots, v_l \in V$, the process contains the parallel composition of $!\overline{p^{\mathrm{V}^l}(id_{v_l}, \ldots, id_{v_0})}\langle true \rangle$, if $v_l \ldots v_0 \in \pi^l(G)$ and $!\overline{p^{\mathrm{V}^l}(id_{v_l}, \ldots, id_{v_0})}\langle false \rangle$ otherwise. This way a node x_0 can check the validity of a path $x_l \ldots x_0$ of the length l via the private channel $p^{\mathrm{V}^l}(x_l, \ldots, x_0)$.

– Select chooses k nodes on a valid path non-deterministically. For all nodes $v_0, \ldots, v_k \in V$, such that $v_0 \ldots v_k \in \pi^k(G)$, the process contains the parallel composition of $!\overline{p^{\mathrm{S}}(id_{v_0})}\langle (id_{v_1}, \ldots, id_{v_k}) \rangle$. This way, a node x can receive a nondeterministic choice of k nodes on a valid path starting in x via the private channel $p^{\mathrm{S}}(x)$.
– $\mathsf{Select}^\star$ chooses a next node which follows k nodes on a valid path non-deterministically. For all nodes $v_0, \ldots, v_k \in V$, such that $v_0 \ldots v_k \in \pi^k(G)$, the process contains the parallel composition of $!\overline{p^{\mathrm{S}^\star}(id_{v_0}, \ldots, id_{v_{k-1}})}\langle id_{v_k} \rangle$. A node x_0 can receive a nondeterministic choice of a next node which follows k nodes $x_0, \ldots, x_{k-1}$ on valid path via the private channel $p^{\mathrm{S}^\star}(x_0, \ldots, x_{k-1})$.
– Key provides access to established shared keys. In the beginning of the process $\mathsf{Network}(G, C, k)$, exactly one key is generated for each pair of nodes when the distance between them is up to k. For each key K_i shared between nodes $u, v \in V$, such that the distance between them is up to k, 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 $p^{\mathrm{K}}(x, y)$.
– $\mathsf{Channel}$ 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 obtains its identification id and the name of channel c for receiving messages via the private channel p^{Top} (line 1). Then, the process consists of the parallel composition of processes:

– Message initiation (lines 2–5). The process creates a new message m and marks the initiation of the message in the event $\mathsf{INIT}(m, id)$ (line 3). Via private channels, it receives nondeterministic selections of nodes $A_1, \ldots, A_k$ on the valid path $idA_1 \ldots A_k$, corresponding keys $K_1, \ldots, K_k$ individually shared with selected nodes, and the communication channel c_1 of the next hop A_1. Finally, it sends the routed message R_1 on the channel c_1 (line 5).

- i-hop after initiation (lines 6–11), where $i < k$. The process receives a routed message R_i on its channel c from a sender A_{i-1}. It checks whether the identification of a node A_0 is correctly included in the received message as an originator of the message, i. e., $A_0 = o$ and the routed message is designated for the node, i. e., $id = A_i$. A next-hop in the route is assigned in the variable A_{i+1} (line 7). The node checks whether the path $A_0 \dots A_i$ is a valid path of the length i (line 8). Then, the process obtains keys K_j individually shared with all previous hops A_j, where $j \in \{0, \dots, i-1\}$. It computes the chain of MACs $H((o, m), \langle K_0, \dots, K_{i-1} \rangle)$ and compares it with received h_i. If the test succeeds, it marks the acceptation of the message m originated from the node o in the event $\mathsf{ACCEPT}(m, o, id)$ (line 10). The process chooses the next-hop A_{i+k}, which follows the last already chosen hop A_{k+i-1} in the route. Then, it obtains keys K_j individually shared with all next nodes A_j, where $j \in \{i+1, \dots, k+i\}$. It retrieves a name of the channel c_{i+1} of the next-hop A_{i+1} and sends the routed message R_{i+1} on the channel c_{i+1} (line 11). Note that the depth of the chain of MACs designated for nodes $A_{i+1}, \dots, A_k$ has the same value $i + 1$. The depth of the chain of MACs designated for nodes $A_k, \dots, A_{k+i}$ decreases from $i + 1$ to 1.
- Normal routing (lines 12–17). The node receives a routed message R_k on its channel c from a sender A_{-1}. The received message R_k contains an original message (o, m) from the originator o (line 13). The process checks whether the routed message is designated for the node, i. e., $id = A_0$, and the path $A_{-k} \dots A_0$ is valid (line 14). Then, the process receives keys K_j individually shared with all previous hops A_j, where $j \in \{-k, \dots, -1\}$. It computes the chain of MACs $H((o, m), \langle K_{-k}, \dots, K_{-1} \rangle)$ and compares it with received h_0 (line 15). If the test succeeds, it marks the acceptation of the message m originated from the node o in the event $\mathsf{ACCEPT}(m, o, id)$ (line 16). Then, the process chooses the next hop A_k, which follows the last hop in the route A_{k-1} (line 16). Then, it obtains keys K_j individually shared with the next hops A_j, where $j \in \{1, \dots, k\}$ (line 16). It retrieves the channel c_1 of the next-hop A_1 and sends the routed message R_k on the channel c_1 (line 17). Note that the depth of the chain of MACs designated for nodes $A_1, \dots, A_k$ decreases from k to 1.

5 A Formal Analysis of the Data 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 $\mathcal{T}' \preceq \mathcal{T}$ hold:*

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

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.

$Node \triangleq$

1. $p^{\mathrm{Top}}(id, c)$.

2. (*//Message initiation*
3. $!\nu m$. **event** $\mathsf{INIT}(m, id)$.
4. $p^{\mathrm{S}}(id)((A_1, \ldots, A_k))$. $\sum_{i=1}^{k} p^{\mathrm{K}}(id, A_i)(K_i)$. $p^{\mathrm{Ch}}(A_1)(c_1)$.
5. $\overline{c_1}\langle R_1((id, m), id, A_1, \ldots, A_k, h((\mathbf{1}, (id, m)), K_1), \ldots, h((\mathbf{1}, (id, m)), K_k))\rangle$) |

6. $\prod_{i=1}^{k-1}$(*//i-hop after initiation*
7. $!c(R_i((o, m), A_0, \ldots, A_{k+i-1}, h_i, \ldots, h_{k+i-1}))$.
8. **if** $o = A_0$ **then if** $id = A_i$ **then** $p^{\mathrm{V}^i}(A_0, \ldots, A_i)(x)$. **if** $x = true$ **then**
9. $\sum_{j=0}^{i-1} p^{\mathrm{K}}(id, A_j)(K_j)$. **if** $h_i = H((o, m), \langle K_0, \ldots, K_{i-1}\rangle)$ **then**
10. **event** $\mathsf{ACCEPT}(m, o, id)$. $p^{\mathrm{S}^\star}(A_i, \ldots, A_{k+i-1})(A_{k+i})$. $\sum_{j=i+1}^{k+i} p^{\mathrm{K}}(id, A_j)(K_j)$.
11. $p^{\mathrm{Ch}}(A_{i+1})(c_{i+1})$. $\overline{c_{i+1}}\langle R_{i+1}((o, m), A_0, \ldots, A_{i+k}, h((\mathbf{i+1}, ((o, m), h_{i+1})), K_{i+1}), \ldots,$
    $h((\mathbf{i+1}, ((o, m), h_k)), K_k), \ldots, h((\mathbf{1}, (o, m)), K_{k+i}))\rangle$) |

12. (*//Normal routing*
13. $!c(R_k((o, m), A_{-k}, \ldots, A_{k-1}, h_0, \ldots, h_{k-1}))$.
14. **if** $id = A_0$ **then** $p^{\mathrm{V}^k}(A_{-k}, \ldots, A_0)(x)$. **if** $x = true$ **then**
15. $\sum_{j=-k}^{-1} p^{\mathrm{K}}(id, A_j)(K_j)$. **if** $h_0 = H((o, m), \langle K_{-k}, \ldots, K_{-1}\rangle)$ **then**
16. **event** $\mathsf{ACCEPT}(m, o, id)$. $p^{\mathrm{S}^\star}(A_0, \ldots, A_{k-1})(A_k)$. $\sum_{j=1}^{k} p^{\mathrm{K}}(id, A_j)(K_j)$.
17. $p^{\mathrm{Ch}}(A_1)(c_1)$. $\overline{c_1}\langle R_k((o, m), A_{-k+1}, \ldots, A_k, h((\mathbf{k}, ((o, m), h_1)), K_1), \ldots,$
    $h((\mathbf{2}, ((o, m), h_{k-1})), K_{k-1}), h((\mathbf{1}, (o, m)), K_k))\rangle$)

Fig. 3. The process Node

Lemma 2. *For an arbitrary communication graph G, a set of captured nodes C, a security parameter $k \geq 2$ and for all traces $\mathcal{T}$ of the corresponding process $\mathsf{Network}(G, C, k)$ in the presence of an active adversary,*

- $\mathcal{T} \vDash^{\mathrm{m}} (c, m')$ *and* $\mathcal{T} \vDash^{\mathrm{m}} (p^{\mathrm{K}}(r, s), K)$,
- $F^{R_i}_{k+i+2}(m') = H((o, m), \langle K_1, \ldots, K_i\rangle)$, *where* $i \leq k$ *and* $K \in \{K_1, \ldots, K_i\}$,
- $\mathcal{T} \vDash^{\mathrm{e}} \mathsf{HONEST}(r)$ *and* $\mathcal{T} \vDash^{\mathrm{e}} \mathsf{HONEST}(s)$, *imply that*

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

Proof. A message m' of the form R_i, where $i \in \{1, \ldots, k\}$, was sent on some channel c. It contains a term $H((o, m), \langle K_1, \ldots, K_i\rangle)$, where $K = K_j$ for some $j \in \{1, \ldots, i\}$. The key K_j is shared between two honest nodes r, s. With respect to the definition of the chain of message authentication codes and Lemma 1, the occurred term $H((o, m), \langle K_1, \ldots, K_i\rangle)$ is uniquely created from the term $H((o, m), \langle K_1, \ldots, K_j\rangle)$. According to the definition of $\mathsf{Network}(G, C, k)$, the key K_j is not published on the "attacker" channel c_{A}. With respect to the definition of the process Node, creating and sending of the term $H((o, m), \langle K_1, \ldots, K_j\rangle)$ (Fig. 3 lines 5, 11, 17) by nodes r, s can be done after marking corresponding events. The term $H((o, m), \langle K_1, \ldots, K_j\rangle)$ could be created by honest nodes r, s:

- in the message initiation (Fig. 3 line 5). Then $i = j = 1$, ($r = o$, or $s = o$) and the event $\mathsf{INIT}(m, o)$ must be executed.
- in the message routing (Fig. 3 lines 11, 17). Then, the event $\mathsf{ACCEPT}(m, o, s)$, or the event $\mathsf{ACCEPT}(m, o, r)$ must be executed.

Let us assume that not one 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 $H((o, m), \langle K_1, \ldots, K_j \rangle)$ must be created by the attacker. For this purpose, the key K_j must be in his knowledge. On the other hand, the key K_j 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_j from eavesdropping the communication in the trace $\mathcal{T}$, because the key K_j can be sent by honest nodes r, s (Fig. 3 lines 5, 11, 17) only in a term $h(\_, K_j)$ 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 1. *For an arbitrary communication graph G, a set of captured nodes C, and a security parameter $k \geq 2$, assuming that on each path of the length $k-1$ in G there exists at least one honest node, it holds that all traces $\mathcal{T}$ of the corresponding process $\mathsf{Network}(G, C, k)$ in the presence of an active adversary satisfy the data integrity property.*

Proof. We prove the theorem by contradiction. Let us assume that the statement is not true. Therefore, there exists a communication graph G', a set of captured nodes C', and a parameter $k \geq 2$, such that on each path of the length $k-1$ in G' there exists at least one honest node. There also exists a closed process that represents the active adversary interacting with the process $\mathsf{Network}(G', C', k)$. 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 exist m', o', id' 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 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 $q \leq n$. 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 $\mathcal{T}''' \vDash^{\mathrm{e}} \mathsf{ACCEPT}(m', o', x)$. From the trace $\mathcal{T}'$, it holds $\mathcal{T}''' \nvDash^{\mathrm{e}} \mathsf{INIT}(m', o')$ and $\mathcal{T}''' \nvDash^{\mathrm{e}} \mathsf{CAPTURED}(o')$. Intuitively, the honest node id'' is the first honest node that accepts the message m' originated from o' exactly at q-th 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 a channel name c'' on the private channel p^{Top}. More formally, there exists $c'' \in E'_{q-1}$ such that $\mathcal{T}''' \vDash^{\mathrm{m}} (p^{\mathrm{Top}}, (id'', c''))$.

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

- i-hop after initiation, where $i \leq k-1$. According to the definition of the appropriate part of the process Node (Fig. 3 lines 6–9), there must exist $K''_0, \ldots, K''_{i-1} \in E'_{q-1}$, such that holds $\mathcal{T}''' \vDash^{\mathrm{m}} (p^{\mathrm{K}}(id'', o'), K''_0)$ and $\mathcal{T}''' \vDash^{\mathrm{m}} (c'', R_i((o', m'), o', \ldots, H((o', m'), \langle K''_0, \ldots, K''_{i-1} \rangle), \ldots))$. With respect to the

definition of the process Topology from $\mathcal{T}''' \not\models^{\mathrm{e}} \mathsf{CAPTURED}(o')$ we obtain $\mathcal{T}''' \models^{\mathrm{e}} \mathsf{HONEST}(o')$. However, together with $\mathcal{T}''' \not\models^{\mathrm{e}} \mathsf{ACCEPT}(m', o', o')$, $\mathcal{T}''' \not\models^{\mathrm{e}} \mathsf{ACCEPT}(m', o', id'')$, $\mathcal{T}''' \not\models^{\mathrm{e}} \mathsf{INIT}(m', o')$, this contradicts Lemma 2.

– Normal routing. According to the definition of the process Node (Fig. 3 lines 13–15), there must exist $K''_{-k}, \ldots, K''_{-1}, A''_{-k}, \ldots, A''_{-1} \in E'_{q-1}$, such that holds $\mathcal{T}''' \models^{\mathrm{m}} (p^{\mathrm{K}}(id'', A''_j), K''_j)$, for all $j \in \{-k, \ldots, -1\}$ and $\mathcal{T}''' \models^{\mathrm{m}} (c'', R_k((o', m'), A''_{-k}, \ldots, A''_{-1}, id'', \ldots, H((o', m'), \langle K_{-k}, \ldots, K_{-1}\rangle), \ldots))$. We consider all possibilities according to the honesty of previous hops $A''_{-k}, \ldots, A''_{-1}$ in the route:
  - All previous hops are captured, i. e., $\mathcal{T}''' \models^{\mathrm{e}} \mathsf{CAPTURED}(A''_j)$ for all $j \in \{-k, \ldots, -1\}$. With respect to the definition of the process Node (Fig. 3 line 14), we have $T''' \models^{\mathrm{m}} (p^{V^k}(A''_{-k}, \ldots, A''_{-1}, id''), \mathit{true})$. According to the definition of the process Topology, the path $A''_{-k} \ldots A''_{-1} id''$ is the valid path of the length k. Nevertheless, the existence of the valid path $A''_{-k} \ldots A''_{-1}$ of the length $k-1$ which consists of all captured nodes contradicts the assumption.
  - At least one previous hop A''_j is honest, i. e., $\mathcal{T}''' \models^{\mathrm{e}} \mathsf{HONEST}(A''_j)$, for some $j \in \{-k, \ldots, -1\}$. Nevertheless, facts $\mathcal{T}''' \not\models^{\mathrm{e}} \mathsf{ACCEPT}(m', o', id'')$, $\mathcal{T}''' \not\models^{\mathrm{e}} \mathsf{ACCEPT}(m', o', A''_j)$, and $\mathcal{T}''' \not\models^{\mathrm{e}} \mathsf{INIT}(m', o')$ contradict Lemma 2.

6 Conclusion

In the paper we presented design and analysis of a k-generalized Canvas protocol. We discussed a motivation for generalization of the Canvas protocol and proposed a k-generalized version of the scheme. The design of the scheme combines the property of the communication graph of the sensor network with the used message authentication codes. In particular, for the case $k = 2$ we improved the Canvas scheme, since in the routed message only two authenticators are sent, fewer than three in the original scheme. Similar problem was studied in [12] where Zhu et al. proposed the scheme which protects against false data injection attacks. In future work we would like to consider application of the proposed k-generalized Canvas scheme for this and similar problems stated in literature.

We built a formal model of the k-generalized 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. Development of this kind of model was formulated as a challenge in the computer security research field [5]. In the semantic model of the applied pi-calculus we specified the data integrity property of the scheme. Finally, we proved that proposed k-generalized Canvas scheme in the presence of an active adversary provides the data integrity of messages. We proved the property for a security parameter $k \geq 2$, an arbitrary communication graph and a set of captured nodes assuming that at least one honest node exists on each path of the length $k-1$ in the communication graph. The proposed formal model could be applied to other WSN security protocols as well. In future work we are planning to analyze other WSN security protocols focusing on different phases of the sensor network in order to model

the abilities of an adversary in more detail. The applied pi-calculus allows us to model various cryptographic primitives. Moreover, the Proverif tool [4] can be used for automatical verification of correspondences [3]. Unfortunately, an automatical analysis could be done for a concrete communication graph and a set of captured nodes.

References

1. Abadi, M., Fournet, C.: Mobile values, new names, and secure communication. In: Proceedings of the 28th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, pp. 104–115. ACM, New York (2001)
2. Armando, A., et al.: The Avispa Tool for the automated validation of internet security protocols and applications. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 281–285. Springer, Heidelberg (2005)
3. Blanchet, B.: Automatic Verification of Correspondences for Security Protocols. Journal of Computer Security 17(4), 363–434 (2009)
4. Blanchet, B.: An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In: Proceedings of 14th IEEE Computer Security Foundations Workshop, pp. 82–96. IEEE Computer Society, Washington (2001)
5. Gollmann, D.: Protocol analysis for concrete environments. In: Moreno Díaz, R., Pichler, F., Quesada Arencibia, A. (eds.) EUROCAST 2005. LNCS, vol. 3643, pp. 365–372. Springer, Heidelberg (2005)
6. Menezes, A., van Oorshot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
7. Tobarra, L., et al.: Model Checking Wireless Sensor Network Security Protocols: TinySec + LEAP. In: Orozco-Barbosa, L., et al. (eds.) Wireless Sensor and Actor Networks. IFIP, vol. 248, pp. 95–106. Springer, Boston (2008)
8. Vogt, H.: Exploring Message Authentication in Sensor Networks. In: Castelluccia, C., Hartenstein, H., Paar, C., Westhoff, D. (eds.) ESAS 2004. LNCS, vol. 3313, pp. 19–30. Springer, Heidelberg (2005)
9. Vogt, H.: Increasing Attack Resiliency of Wireless Ad Hoc and Sensor Networks. In: Proceedings of the Second International Workshop on Security in Distributed Computing Systems (ICDCSW 2005), vol. 2, pp. 179–184. IEEE Computer Society, Washington (2005)
10. Vogt, H.: Integrity preservation for communication in sensor networks. Technical Report 434, Institute for Pervasive Computing. ETH Zürich (2004)
11. Xiao, Y., Du, X.: A Survey on Sensor Network Security. In: Li, Y., et al. (eds.) Wireless Sensor Networks and Applications. Signals and Communication Technology Series, pp. 403–421. Springer, Heidelberg (2008)
12. Zhu, S., Setia, S., Jajodia, S., Ning, P.: Interleaved hop-by-hop authentication against false data injection attacks in sensor networks. ACM Transactions on Sensor Networks 3(3), article 14 (2007)