Journal of Automata, Languages and Combinatorics **11** (2006) 4, 345–374

A Team Automaton Scenario for the Analysis of Security Properties of Communication Protocols [1]

Maurice H. ter Beek
ISTI–CNR, via G. Moruzzi 1, 56124 Pisa, Italy
e-mail: maurice.terbeek@isti.cnr.it

Gabriele Lenzini
Telematica Instituut, P.O. Box 589, 7500 AN Enschede, The Netherlands
e-mail: gabriele.lenzini@telin.nl

and

Marinella Petrocchi
IIT–CNR, via G. Moruzzi 1, 56124 Pisa, Italy
e-mail: marinella.petrocchi@iit.cnr.it

ABSTRACT

Formal methods are a popular means to specify and verify security properties of a variety of communication protocols. In this article we take a step towards the use of team automata for the analysis of security aspects in such protocols. To this aim, we define an insecure communication scenario for team automata that is general enough to encompass various communication protocols. We then reformulate the Generalized Non-Deducibility on Compositions schema—originally introduced in the context of process algebrae—in terms of team automata. Based on the resulting team automata framework, we subsequently develop two analysis strategies that can be used to verify security properties of communication protocols. Indeed, the paper concludes with two case studies in which we show how our framework can be used to prove integrity and secrecy in two different settings: We show how integrity is guaranteed in a team automaton model of a particular instance of the Efficient Multi-chained Stream Signature protocol, a communication protocol for signing digital streams that provides some robustness against packet loss, and we show how secrecy is preserved when a member of a multicast group leaves the group in a particular run of the complementary variable approach to the N-Root/Leaf pairwise keys protocol.

Keywords: Team automata; Security analysis; Security properties; Communication protocols

[1] The work presented here was partly supported by the EU project IST-3-016004-IP-09 Sensoria (*Software Engineering for Service-Oriented Overlay Computers*).

1. Introduction

Recent years have seen an increasing interest in the use of automata-based formalisms to specify and verify security properties of communication protocols [28, 30, 31, 32, 36, 40, 41]. We contribute to this line of research by providing a study into the use of team automata for this purpose.

Team automata form a flexible formal model to capture notions like communication, coordination, cooperation and collaboration in reactive, distributed systems. They allow one to separately specify the components of such a system and to consequently describe their interactions through synchronizations of shared actions. Technically, team automata are an extension of I/O automata [37] and in [5] it is shown how I/O automata fit in the framework of team automata. While originally introduced in the context of Computer Supported Cooperative Work to formalize the conceptual and architectural levels of groupware systems [3, 16, 29], team automata have since enjoyed increasing popularity in the context of computer security [1, 2, 6, 7, 8, 15, 33, 43]. We now briefly describe the content of each of these publications. In [2] various access control strategies have been specified and analyzed by team automata. In [6, 7] it was shown that team automata allow a very natural way of modelling secure multicast and broadcast communication protocols. Also contained in [6, 7] is a specification by team automata of an instance of the Efficient Multi-chained Stream Signature (EMSS) protocol [42] and an initial attempt to reformulate the Generalized Non-Deducibility on Compositions (GNDC) schema—originally introduced in the context of process algebrae [23]—in terms of team automata. In [15] an effort was made to use team automata to model and analyze a privacy property of a protocol by Cachin *et al.* [10] to secure mobile agents in a hostile environment. Finally, in [1, 8, 33, 43] the use of team automata in the context of computer security analysis is discussed in a broader context.

In this article we study a general team automata framework for the analysis of security issues of communication protocols. First we define an insecure communication scenario for team automata by adding a most general intruder *à la* Dolev-Yao [14] to a team automaton model of a secure communication protocol. This insecure scenario is general enough to encompass various communication protocols. Second, we reformulate the GNDC schema in terms of team automata and subsequently describe two analysis strategies for the insecure scenario, which can be used to verify those security properties that can be expressed in the GNDC schema. Third, we apply this framework by showing that integrity is guaranteed in a case study in which team automata model a particular instance of the EMSS protocol and that secrecy is preserved in a case study in which team automata model a particular run of the complementary variable approach to the N-Root/Leaf pairwise keys protocol. The aim of the two case studies is not in the first place to provide new insights into the involved protocols, but rather to illustrate our approach for two well-known protocols, thus allowing an easy comparison for those familiar with other approaches. To summarize, we provide a solid theoretical basis for the use of team automata to analyze security aspects of communication protocols. While the two small case studies demonstrate the potential applicability of our framework, we are aware that its practical success stands or falls

with the availability of a tool in which the user can specify and automatically verify security properties of communication protocols modelled by team automata. This is a goal for the future.

As said before, our automata-based approach to the analysis of security properties is not unique. We now briefly describe some of the most closely related automata-based approaches. In [36], an experiment involving the combination of simple shared-key communication with the Diffie-Hellman key distribution protocol [13] is modelled and proved correct using I/O automata—recall that team automata are an extension of I/O automata [5]. As noted by the author herself, a limitation of this approach is the fact that the protocol only allows purely passive eavesdroppers to listen in on the communication. With respect to our approach, this choice simplifies the formulation of compositional results since an eavesdropper cannot change the course of communication, *e.g.,* by conducting a communication in which it pretends to be an honest participant. Her approach does provide attractive compositional reasoning techniques. In [40, 41], interacting state machines—another extension of I/O automata—are applied to the analysis of security properties. They are used to model and analyze the classic Needham-Schroeder public-key authentication protocol in the version fixed by Lowe [35]. A strong point of this approach is the machine assistance to define and verify interacting state machines. What is missing with respect to our approach, however, are solid techniques for compositional reasoning over more complex communication protocols. In [28], asynchronous product automata are used to specify and verify cryptographic communication protocols, in particular the classic symmetric Needham-Schroeder protocol [39]. These automata differ from team automata in that they are a shared-memory model of communicating automata. A big plus of their approach is the tool support. In [30, 31, 32], probabilistic timed automata and parametric probabilistic transition systems are applied to model and analyze communication protocols, among which the non-repudiation protocol. Furthermore, the Non-Deducibility on Compositions (NDC) schema—originally introduced in the context of process algebrae [17]—is extended with time and probability. While this approach still lacks a tool and while automata with time and probability inevitably become more complex to deal with, taking into account time and probability obviously makes the analysis of security properties in communication protocols more realistic [26]—which is thus an advantage over our approach. We, however, provide a static characterization of the intruder in our GNDC schema in terms of team automata in Section 4, thus circumventing the universal quantification. This is not always easy—or even feasible—as may be concluded from [32], where a Probabilistic Timed NDC schema which contains a universal quantification is introduced.

This article is organized as follows. In Section 2 we define team automata, after which we describe an insecure communication scenario for team automata in Section 3. In Section 4 we reformulate the GNDC schema in terms of team automata and we enrich the insecure scenario with two analysis strategies. Subsequently we apply these strategies in Section 5 by verifying integrity and secrecy in two different case studies. Finally, we conclude with a summary of the main results and some directions for future work.

2. Team Automata

A team automaton consists of component automata—ordinary automata without final states and with their alphabet partitioned into input, output and internal actions—combined in a coordinated way such that they can perform shared actions. Internal actions have strictly local visibility and cannot be used for communication with other component automata, while input and output actions together form the external actions that are observable by other components and that are used for communication between components. During each clock tick the components within a team can simultaneously participate in one instantaneous action, *i.e.,* synchronize on this action, or remain idle. Component automata can thus be combined in a loose or more tight fashion depending on which actions are to be synchronized, and when. Team automata can in turn be used as components in a higher-level team automaton.

We now fix some notations and terminology used throughout this article, after which we recall some definitions and results concerning team automata from [1, 3, 4].

For convenience we sometimes denote the set $\{1, \ldots, n\}$ by $[n]$; thus $[0] = \varnothing$. The (cartesian) product of sets V_i, with $i \in [n]$, is denoted by $\prod_{i\in[n]} V_i$. In addition to the prefix notation, we also use the infix notation $V_1 \times \cdots \times V_n$. For $j \in [n]$, $\mathrm{proj}_j : \prod_{i\in[n]} V_i \to V_j$ is defined by $\mathrm{proj}_j((a_1, \ldots, a_n)) = a_j$. The powerset of a set V is denoted by 2^V.

Let Σ and Γ be sets of symbols. The morphism $\mathrm{pres}_{\Sigma,\Gamma} : \Sigma^* \to \Gamma^*$ is defined as follows. For any $\sigma \in \Sigma^*$, $\mathrm{pres}_{\Sigma,\Gamma}(\sigma x) = \mathrm{pres}_{\Sigma,\Gamma}(\sigma)x$ if $x \in \Gamma$ and $\mathrm{pres}_{\Sigma,\Gamma}(\sigma x) = \mathrm{pres}_{\Sigma,\Gamma}(\sigma)$ otherwise. It thus preserves the symbols from Γ and erases all other symbols. We omit Σ when no confusion can arise.

Let $f : A \to A'$ and $g : B \to B'$ be functions. Then $f \times g : A \times B \to A' \times B'$ is defined as $(f \times g)(a, b) = (f(a), g(b))$. We use $f^{[2]}$ as shorthand for $f \times f$.

Definition 1 *A* component automaton *is a* $\mathcal{C} = (Q, (\Sigma_{inp}, \Sigma_{out}, \Sigma_{int}), \delta, I)$*, with set* Q *of* states*; set* $\Sigma = \Sigma_{inp} \cup \Sigma_{out} \cup \Sigma_{int}$ *of* actions *specified by three pairwise disjoint sets* Σ_{inp}*,* Σ_{out} *and* Σ_{int} *of* input*,* output *and* internal *actions, resp., and such that* $Q \cap \Sigma = \varnothing$*; set* $\delta \subseteq Q \times \Sigma \times Q$ *of* (labelled) transitions*; and set* $I \subseteq Q$ *of* initial states*.*

The set $\mathbf{C}_\mathcal{C}$ *of* computations *of* $\mathcal{C}$ *is defined as* $\mathbf{C}_\mathcal{C} = \{\, q_0 a_1 q_1 \cdots a_n q_n \mid n \geq 0$ *and* $(q_{i-1}, a_i, q_i) \in \delta$ *for all* $i \in [n] \,\}$*.*

The Γ-behaviour $\mathbf{B}^\Gamma_\mathcal{C}$ *of* $\mathcal{C}$*, with* $\Gamma \subseteq \Sigma$*, is defined as* $\mathbf{B}^\Gamma_\mathcal{C} = \{\, pres_\Gamma(\alpha) \mid \alpha \in \mathbf{C}_\mathcal{C} \,\}$*.*

The Σ-behaviour $\mathbf{B}^\Sigma_\mathcal{C}$ of $\mathcal{C}$ is also called the *behaviour* of $\mathcal{C}$, in which case Σ may be omitted from the notation. We let Σ_{ext} and Σ_{loc}, resp., denote the set $\Sigma_{inp} \cup \Sigma_{out}$ of *external* and the set $\Sigma_{out} \cup \Sigma_{int}$ of *locally-controlled* actions of $\mathcal{C}$; its Σ_{ext}-behaviour $\mathbf{B}^{\Sigma_{ext}}_\mathcal{C}$ and its Σ_{loc}-behaviour $\mathbf{B}^{\Sigma_{loc}}_\mathcal{C}$, resp., are also called its *external behaviour* and its *locally-controlled behaviour*. Let $a \in \Sigma$. The set δ_a of *a-transitions* of $\mathcal{C}$ is defined as $\delta_a = \{\, (q, q') \mid (q, a, q') \in \delta \,\}$. Finally, note that behavioural inclusion defines a preorder relation on automata.

For the sequel we let $\mathcal{S} = \{\, \mathcal{C}_i \mid i \in [n] \,\}$, for some $n \geq 1$, be an arbitrary but fixed set of component automata specified as $\mathcal{C}_i = (Q_i, (\Sigma_{i,inp}, \Sigma_{i,out}, \Sigma_{i,int}), \delta_i, I_i)$, with set $\Sigma_i = \Sigma_{i,inp} \cup \Sigma_{i,out} \cup \Sigma_{i,int}$ of actions, set $\Sigma_{i,ext} = \Sigma_{i,inp} \cup \Sigma_{i,out}$ of external

actions, and set $\Sigma_{i,loc} = \Sigma_{i,out} \cup \Sigma_{i,int}$ of locally-controlled actions. Furthermore, we let $\Sigma = \bigcup_{i\in[n]} \Sigma_i$.

When composing team automata over $\mathcal{S}$, the internal actions of the components constituting $\mathcal{S}$ must be private, *i.e.,* uniquely associated to one component automaton. This is formally expressed by requiring that $\Sigma_{i,int} \cap \bigcup_{j\in([n]-\{i\})} \Sigma_j = \varnothing$, for all $i \in [n]$, *i.e.,* no internal action of any component from $\mathcal{S}$ may appear as an action in any of the other components constituting $\mathcal{S}$ (recall that a component's input, output and internal actions are pairwise disjoint). If this is the case, then $\mathcal{S}$ is called a *composable system* and for the sequel we let $\mathcal{S}$ be a composable system.

The state space of a team automaton composed over $\mathcal{S}$ is the product of the state spaces of the components constituting $\mathcal{S}$. Also the set of actions of a team over $\mathcal{S}$ is uniquely determined, *viz.* as follows. The internal actions of the components are the internal actions of the team. Each action which is output for one or more of the components is an output action of the team. In particular, an action that is an output action of one component and also an input action of another component, is considered an output action of the team. The input actions of the team that do not occur at all as output actions of any of the components constituting $\mathcal{S}$, are the input actions of the team. The transitions of a team over $\mathcal{S}$, finally, are not fixed by the transitions of the components constituting $\mathcal{S}$. Instead they are chosen by allowing certain *synchronizations* on actions, while excluding others.

Definition 2 *Let $a \in \Sigma$. The set $\Delta_a(\mathcal{S})$ of* synchronizations *of a in $\mathcal{S}$ is defined as* $\Delta_a(\mathcal{S}) = \{\, (q,q') \in \prod_{i\in[n]} Q_i \times \prod_{i\in[n]} Q_i \mid (\, \exists\, j \in [n] :\ proj_j{}^{[2]}(q,q') \in \delta_{j,a}\,)$ *and* $(\, \forall\, i \in [n] :\ (\, proj_i{}^{[2]}(q,q') \in \delta_{i,a}\,)$ *or* $(\, proj_i(q) = proj_i(q')\,)\,)\,\}$.

The set $\Delta_a(\mathcal{S})$ thus contains all possible combinations of a-transitions of the components constituting $\mathcal{S}$, with all non-participating components remaining idle. It is explicitly required that in every synchronization at least one component participates. The state change of a team automaton over $\mathcal{S}$ is thus defined by the local state changes of the components constituting $\mathcal{S}$ that participate in the action of the team being executed. When defining a team automaton over $\mathcal{S}$, a specific subset of $\Delta_a(\mathcal{S})$ must thus be chosen for each action a. This defines a certain kind of communication between the components constituting the team.

Definition 3 *A* team automaton *over $\mathcal{S}$ is a $\mathcal{T} = (Q, (\Sigma_{inp}, \Sigma_{out}, \Sigma_{int}), \delta, I)$, with states $Q = \prod_{i\in[n]} Q_i$; input actions $\Sigma_{inp} = (\bigcup_{i\in[n]} \Sigma_{i,inp}) - \Sigma_{out}$; output actions $\Sigma_{out} = \bigcup_{i\in[n]} \Sigma_{i,out}$; internal actions $\Sigma_{int} = \bigcup_{i\in[n]} \Sigma_{i,int}$; transitions $\delta \subseteq Q \times \Sigma \times Q$ such that $\delta_a \subseteq \Delta_a(\mathcal{S})$ for all $a \in \Sigma$ and $\delta_a = \Delta_a(\mathcal{S})$ for all $a \in \Sigma_{int}$; and initial states $I = \prod_{i\in[n]} I_i$.*

Each choice of synchronizations thus defines a team automaton. It is trivial to observe that *every team automaton is again a component automaton*, which in its turn can thus be used as a component in an *iteratively* composed team. In this way one can construct, *e.g.,* a team automaton $\mathcal{T}''$ over the composable system $\{\mathcal{T}', \mathcal{C}_3\}$, where $\mathcal{T}'$ is a team composed over the composable system $\{\mathcal{C}_1, \mathcal{C}_2\}$. It may be useful, though,

to *hide* certain external actions of a team automaton before using this team in an iterative composition, in order to prohibit synchronizations on these actions on a higher level of the composition.

Definition 4 *Let* $\mathcal{T} = (Q, (\Sigma_{inp}, \Sigma_{out}, \Sigma_{int}), \delta, I)$ *be a team automaton and let* $\Gamma \subseteq \Sigma_{ext}$*. Then* $hide_\Gamma(\mathcal{T}) = (Q, (\Sigma_{inp} - \Gamma, \Sigma_{out} - \Gamma, \Sigma_{int} \cup \Gamma), \delta, I)$*.*

In $\mathrm{hide}_\Gamma(\mathcal{T})$, the external actions in Γ have thus become unobservable for other automata by regarding them as internal actions. Without formally defining renaming, we assume these actions to be indexed in order to guarantee composability.

It may sometimes be useful to construct a unique team automaton of a specific type. In [3] several strategies for choosing the synchronizations of a team automaton were defined, each leading to a uniquely defined team automaton. These strategies fix the synchronizations of a team automaton by defining—per action a—certain conditions on the a-transitions to be chosen from $\Delta_a(\mathcal{S})$, thus determining a unique subset of $\Delta_a(\mathcal{S})$ as the set of a-transitions of the team. Once such subsets have been chosen for all actions, the team automaton they define is unique. For example, an action a is said to be *action-indispensable* (*ai* for short) in $\mathcal{T}$ if each of its a-transitions is brought about by a synchronization of all components from $\mathcal{S}$ that share a (*i.e.,* have a as an action).

Definition 5 *Let* $a \in \Sigma$*. The set* $\mathcal{R}_a^{ai}(\mathcal{S})$ *is defined as* $\mathcal{R}_a^{ai}(\mathcal{S}) = \{\, (q, q') \in \Delta_a(\mathcal{S}) \mid \forall\, i \in [n] :\ (\, a \in \Sigma_i \Rightarrow proj_i^{[2]}(q, q') \in \delta_{i,a}\,)\,\}$*.*

The set $\mathcal{R}_a^{ai}(\mathcal{S})$ thus contains *all and only* those a-transitions from $\Delta_a(\mathcal{S})$ in which every component automaton with a as an action participates. In particular, the team automaton over $\mathcal{S}$ defined by this set, for all its external actions, is the unique team automaton in which any execution of an external action sees the participation of all components having that action in their set of actions.

Definition 6 $\mathcal{T} = (Q, (\Sigma_{inp}, \Sigma_{out}, \Sigma_{int}), \delta, I)$ *is* the max-ai team automaton *over* $\mathcal{S}$*, denoted by* $|||\,\mathcal{S}$*, if* $\delta_a = \mathcal{R}_a^{ai}(\mathcal{S})$ *for all* $a \in \Sigma$*.*

Remark 1 In [3] it was shown that the behaviour of an iteratively composed max-ai team automaton equals that of the max-ai team automaton over the underlying components, *i.e.,* continuing our above example: If $\mathcal{T}'$ and $\mathcal{T}''$ are the max-ai team automata over $\{\mathcal{C}_1, \mathcal{C}_2\}$ and $\{\mathcal{T}', \mathcal{C}_3\}$, resp., and $\mathcal{T}$ is the max-ai team automaton over $\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$, then $\mathbf{B}_{\mathcal{T}''} = \mathbf{B}_{\mathcal{T}}$.

A team automaton is said to satisfy *compositionality* if its behaviour can be described in terms of that of its constituting component automata, *i.e.,* when the sequences forming the behaviour of a set of component automata can be *shuffled* in such a way that the sequences forming the behaviour of a particular team over these components result. A shuffle of words is an arbitrary interleaving of the symbol occurrences in the original words, like the shuffling of decks of cards. In [4], the shuffle operation was generalized to a variety of *synchronized shuffle* operations: Rather than freely

interleaving the occurrences of the letters in the words being shuffled, some letters may now be subject to "synchronization". This means that occurrences of those letters in different words are combined into one occurrence. The resulting word thus has a "backbone" consisting of occurrences of synchronized letters.

Definition 7 *Let $m \geq 1$ and let Δ_i, with $i \in [m]$, be sets of symbols. The* full synchronized shuffle $\underline{||}_{\{\Delta_i | i \in [m]\}} L_i$ *of $L_i \subseteq \Delta_i^*$, with $i \in [m]$, is defined as* $\underline{||}_{\{\Delta_i | i \in [m]\}} L_i = \{ w \in (\bigcup_{i \in [m]} \Delta_i)^* \mid \forall\, i \in [m] : pres_{\Delta_i}(w) \in L_i \}$.

If we consider, *e.g.,* $L_1 = \{abc, cba\}$ over $\Delta_1 = \{a, b, c\}$ and $L_2 = \{bcd\}$ over $\Delta_2 = \{b, c, d\}$, then $\underline{||}_{\{\Delta_i | i \in [2]\}} L_i = \{abcd\}$. In [4] it was shown that the construction of team automata according to certain types of synchronization guarantees compositionality. Since all synchronizations in a max-ai team automaton require the participation of all its components sharing the action being synchronized, it is not surprising that the behaviour of a max-ai team automaton equals the full synchronized shuffle of the behaviours of its constituting components:

Theorem 1 ([4]) *Let $\mathcal{T}$ be the max-ai team automaton over $\mathcal{S}$. Then* $\mathbf{B}_{\mathcal{T}} = \underline{||}_{\{\Sigma_i | i \in [n]\}} \mathbf{B}_{\mathcal{C}_i}$.

3. An Insecure Communication Scenario for Team Automata

In this section we present an insecure communication scenario for team automata, which we consequently intend to use to analyze security issues of (cryptographic) communication protocols.

We assume all actions to be built over a first-order signature σ, where predicate symbols are seen as communication channels and atomic formulae as messages. We assume the function symbols in σ to contain at least those that we will use in the sequel: Symbols denoting encryption and pairing, *e.g.,* $\{\_\}\_$ and $\langle \_, \_ \rangle$; those denoting hashing, *e.g.,* $h(\_)$; and those indicating the secret and public key, *e.g.,* $sk(\_)$ and $pk(\_)$. We let m, m' range over the set $\mathsf{Messages}$ of atomic formulae and c, c' over the set $\mathsf{Channels}$ of predicate symbols. We will use $\mathtt{Eve}$, $\mathtt{Eve}'$, $\mathtt{Pub}$, $\mathtt{Pub}'$, $\mathtt{Reveal}$ and $\mathtt{Reveal}'$ as particular predicate names. Every action will thus be written as $c(m)$, denoting message m sent over channel c. Given a set $\mathsf{M} \subseteq \mathsf{Messages}$ of messages, we define $c(\mathsf{M}) = \{ c(m) \mid m \in \mathsf{M} \}$. Given a set C of predicate names we define $\mathsf{C}(\mathsf{M}) = \{ c(m) \mid m \in \mathsf{M},\ c \in \mathsf{C} \}$. Finally, with a little abuse of notation, we will also write C as a shortcut for the set $\mathsf{C}(\mathsf{Messages})$.

We abstract from the cryptographic details concerning the operations by which messages can be encrypted, decrypted, hashed, *etc.*, but we assume the presence of a cryptosystem (defined by a single step derivation operator $\vdash$, with $\vdash^*$ as its transitive and reflexive closure) that implements these operations. By applying operations from this cryptosystem to a set M of messages, a new set $\mathsf{D}(\mathsf{M}) = \{ m \mid \mathsf{M} \vdash^* m \}$ of messages (usually called the *deduction set*) can be obtained. This approach is standard in the analysis of (cryptographic) communication protocols [12, 23, 34, 36].

In the sequel we assume a communication protocol specification involving two honest roles, *viz.* an *initiator* $\mathcal{T}_S$ and a *responder* $\mathcal{T}_R$. Rather than a direct communication between them, we assume all their communication to be transmitted over an *insecure connection*. This insecure connection functions as a completely passive pipe that transmits messages between $\mathcal{T}_S$ and $\mathcal{T}_R$ and that may release the transmitted messages to an *intruder* which, in its turn, can either listen to or modify (fake) these messages. When verifying security properties of communication protocols, it is quite common to include an additional intruder (*à la* Dolev-Yao [14]) that is supposed to be malicious and whose aim is to subvert the protocol's correct behaviour. A protocol specification is then considered secure w.r.t. a security property if it satisfies this property despite the presence of the intruder. Based on the approach of [36], the insecure connection and the intruder are modelled by team automata $\mathcal{T}_{IC}$ and $\mathcal{T}_X$. We thus propose a framework of four types of team automata:

1. $\mathcal{T}_S$ plays the role of the protocol's initiator;
2. $\mathcal{T}_R$ plays the role of the protocol's responder;
3. $\mathcal{T}_{IC}$ plays the role of the insecure connection;
4. $\mathcal{T}_X$ plays the role of the active and malicious intruder.

We do not explicitly specify the team automata of our framework, but we informally describe them by their interactions. More precisely, we let the initiator and the responder communicate with the insecure connection through disjoint sets of actions Σ^S_{com} and Σ^R_{com}, resp., such that a direct communication between them is impossible. The insecure connection, in its turn, can interact with the intruder only through a distinct set Σ^I_{com} of actions. Finally, some particular actions Σ^S_{sig} and Σ^R_{sig} may be used by the honest roles in order to reveal some information to the outside concerning, *e.g.,* a state reached during a run of the protocol. In Fig. 1 we have sketched the insecure communication scenario for team automata described above and we have instantiated it with $\Sigma^S_{com} = \{\texttt{Pub}\}$, $\Sigma^R_{com} = \{\texttt{Pub}'\}$, $\Sigma^I_{com} = \{\texttt{Eve}, \texttt{Eve}'\}$, $\Sigma^S_{sig} = \{\texttt{Reveal}\}$ and $\Sigma^R_{sig} = \{\texttt{Reveal}'\}$. Recall that $\{\texttt{Pub}\}$ is a shortcut of $\{\,\texttt{Pub}(m) \mid m \in \mathsf{Messages}\,\}$, *etc.*.

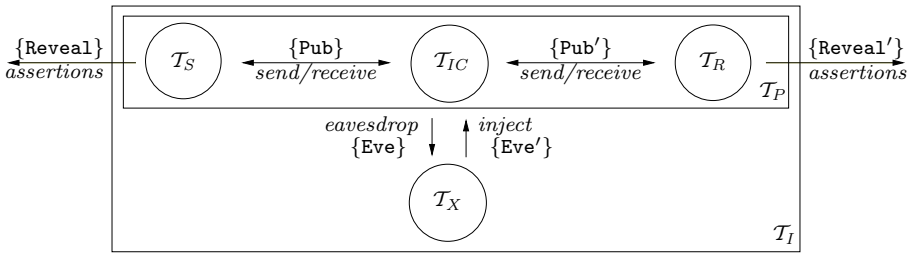

Figure 1: An insecure communication scenario for team automata.

We denote by $\mathcal{T}_P$ the team automaton representing our protocol specification *in the absence of the intruder*. We thus define $\mathcal{T}_P$ to be the max-ai team automaton over

$\{\mathcal{T}_S, \mathcal{T}_R, \mathcal{T}_{IC}\}$ that is obtained after hiding $\Sigma^P_{com} = \Sigma^S_{com} \cup \Sigma^R_{com}$, *i.e.,* all messages transmitted over the insecure connection:

$$\mathcal{T}_P = \text{hide}_{\Sigma^P_{com}}(\,|||\,\{\mathcal{T}_S, \mathcal{T}_R, \mathcal{T}_{IC}\}).$$

By hiding Σ^P_{com}, these actions are no longer available for synchronizations in further team automata composed over $\mathcal{T}_P$. To its environment $\mathcal{T}_P$ thus appears as a black box, with the actions Σ^I_{com}—serving as the backdoor for intrusion—and possibly with some output actions Σ^S_{sig} and Σ^R_{sig}—signalling the successful reception of messages. Usually such signals are used only for verification purposes and for the sequel we assume that $\Sigma^S_{sig} \cap \Sigma^R_{sig} = \varnothing$.

We denote by $\mathcal{T}_I$ the team automaton representing our protocol specification *in the presence of the intruder*. The actions Σ^I_{com} serving as the backdoor for intrusion are exactly what we need to guarantee that the intruder $\mathcal{T}_X$ may communicate with $\mathcal{T}_P$ only by means of the insecure connection. We thus define $\mathcal{T}_I$ to be the max-ai team automaton over $\{\mathcal{T}_P, \mathcal{T}_X\}$ that is obtained after hiding Σ^I_{com}, *i.e.,* all messages that the intruder can eavesdrop from and inject back into the insecure connection:

$$\mathcal{T}_I = \text{hide}_{\Sigma^I_{com}}(\,|||\,\{\mathcal{T}_P, \mathcal{T}_X\}).$$

We have now defined an *insecure* communication scenario for team automata by composing a *secure* communication scenario with an *intruder*.

4. Analyzing Security Properties with Team Automata

In this section we reformulate the Generalized Non-Deducibility on Compositions (GNDC) schema in terms of team automata and we enrich the insecure communication scenario developed in the previous section with two analysis strategies.

4.1. The Generalized Non-Deducibility on Compositions Schema

In the literature several efforts have been made to prevent the unauthorized information flow in multilevel computer systems [9], *i.e.,* systems where processes and objects are bound to a specific security level. An example from the military jargon is the fact that documents are generally hierarchized from unclassified to top secret. The seminal idea of *non interference* proposed in [25] aims to assure that information can only flow from low levels to higher ones. The first taxonomy of non-interference-like properties has been uniformly defined and compared in [17, 18, 19] in the context of a CCS-like process algebra. In particular, processes in the algebra were divided into high and low processes according to the level of actions that they can perform. To detect whether an incorrect information flow (*i.e.,* from high to low) has occurred, a particular non-interference-like property was defined as the so-called *Non Deducibility on Compositions* (NDC). NDC essentially says that a process is secure w.r.t. wrong information flows if its behaviour of low-level processes in isolation appears to be the same as its behaviour of low-level processes when interacting with any high-level process. NDC can be reformulated from the world of multilevel systems to that of network security [21, 23], *viz.* the low-level process becomes a specification of a

(cryptographic) communication protocol and the behaviour of the protocol running in isolation is compared with that of the protocol running in a composition with any possible adversary.

Subsequently, a *Generalized* NDC (GNDC) has been formulated in [23] to encompass in a uniform way many security properties. Informally, the GNDC schema states that a system specification P satisfies property $GNDC^{\alpha,C}_{\triangleleft}$ if the behaviour of P, despite the presence of a hostile environment $\mathcal{E}_C$ that can interact with P only through a fixed set of channels C, *appears* to be the same (w.r.t. a behavioural relation $\triangleleft$ of observational equivalence) as the behaviour of a modified version $\alpha(P)$ of P that represents the *expected* (correct) behaviour of P. The GNDC schema thus has the form

$$P \in GNDC^{\alpha,C}_{\triangleleft} \text{ iff } \forall X \in \mathcal{E}_C : (P \parallel X)\backslash C \triangleleft \alpha(P),$$

where $(P \parallel X)\backslash C$ denotes the parallel composition of processes P and X restricted to communication over channels other than C. X is an arbitrary process in the environment $\mathcal{E}_C$, the set of all processes whose communicating actions are in C. By varying the parameters $\triangleleft$, α and C, the GNDC schema can be used to define and verify many security properties—among which secrecy, integrity and entity authentication [17, 20, 23, 27, 38].

In the specific context of analyzing (cryptographic) communication protocols, the *static* (initial) knowledge of the hostile environment must be bound to a specific set of messages. This is needed to avoid a hostile intruder that would be too strong and able to corrupt any secret as it would know all (cryptographic) keys, *etc.*. This brings us to the definition of a new environment $\mathcal{E}^{\phi}_C$, based on $\mathcal{E}_C$, of all processes communicating through channels C and having an initial knowledge of at most the messages in $\mathsf{D}(\phi)$. For the analysis of safety properties (such as secrecy, integrity and entity authentication) it suffices to consider the trace inclusion relation $\leq$ as behavioural relation between the terms of the algebra [23]. Hence we consider the GNDC instance

$$P \in GNDC^{\alpha,C}_{\leq} \text{ iff } \forall X \in \mathcal{E}^{\phi}_C : (P \parallel X)\backslash C \leq \alpha(P), \tag{1}$$

which was, *e.g.,* used in [27] to analyze integrity in stream signature protocols. Informally, (1) requires traces of process $(P \parallel X)\backslash C$ to be included in the traces of process $\alpha(P)$, representing the expected behaviour of P when no adversary is present.

4.2. Reformulating GNDC in Terms of Team Automata

We now reformulate the GNDC schema in terms of team automata. We first instantiate P to be a team automaton modelling communication between an initiator and a set of responders over an insecure connection, in the style of the team automaton $\mathcal{T}_P$ considered in the insecure communication scenario of Section 3. Since (1) requires P to communicate with X through the channels contained in C, we define a set $\mathsf{C} = \mathsf{C}_{inp} \cup \mathsf{C}_{out}$ of actions C_{inp} that are input to $\mathcal{T}_X$ and actions C_{out} that are output by $\mathcal{T}_X$. Moreover, we require the actions in C to be external actions of $\mathcal{T}_P$ different from those in Σ^P_{com}, Σ^S_{sig} and Σ^R_{sig}. This is analogous to requiring $\mathcal{T}_P$ to be able to

communicate with the intruder $\mathcal{T}_X$ only by executing actions in Σ^I_{com}. For the sequel we thus assume C to coincide with Σ^I_{com} and, in particular, C_{inp} to coincide with the actions in Σ^I_{com} that are input to $\mathcal{T}_X$ (*e.g.,* $\{\mathtt{Eve}\}$ in Fig. 1) and C_{out} with the actions in Σ^I_{com} that are output by $\mathcal{T}_X$ (*e.g.,* $\{\mathtt{Eve}'\}$ in Fig. 1).

We can now formalize the hostile environment $\mathcal{E}_\mathsf{C}$ in terms of team automata as

$$\mathcal{E}_\mathsf{C} = \{\, (Q, (\Sigma_{inp}, \Sigma_{out}, \Sigma_{int}), \delta, I) \mid \Sigma_{inp} \subseteq \mathsf{C}_{inp},\ \Sigma_{out} \subseteq \mathsf{C}_{out} \,\}. \tag{2}$$

In addition, (1) requires the initial knowledge of the environment to be bound to a specific set of messages ϕ. This informally means that the environment should be able to produce, by means of only its internal functioning, at most the messages contained in $\mathsf{D}(\phi)$. In terms of team automata this means that a component automaton in the environment, when considered as a stand-alone component, can only execute locally-controlled actions belonging to $\mathsf{C}(\mathsf{D}(\phi))$. This is formally defined by restricting its behaviour as far as sequences consisting of solely locally-controlled actions are concerned to sequences over $\mathsf{C}(\mathsf{D}(\phi))$. This suffices because—at a more abstract level—these are the sequences that it can produce without receiving any additional messages from the outside, *i.e.,* by exploiting only its own knowledge. Let $\mathcal{T}$ be a team automaton with locally-controlled actions $\Sigma^{\mathcal{T}}_{loc}$. Then the *initial knowledge* of $\mathcal{T}$ is defined as $\mathbf{B}_\mathcal{T} \cap (\Sigma^{\mathcal{T}}_{loc})^*$, *i.e.,* those sequences of its behaviour consisting of solely locally-controlled actions. The formal definition of the environment $\mathcal{E}^\phi_\mathsf{C}$ in terms of team automata then becomes

$$\mathcal{E}^\phi_\mathsf{C} = \{\, \mathcal{X} \in \mathcal{E}_\mathsf{C} \mid \mathbf{B}_\mathcal{X} \cap (\Sigma^{\mathcal{X}}_{loc})^* \subseteq (\mathsf{C}(\mathsf{D}(\phi)))^* \,\}, \tag{3}$$

in which $\Sigma^{\mathcal{X}}_{loc}$ denotes the locally-controlled actions of $\mathcal{X}$.

Finally we need a behavioural notion of comparison between team automata which abstracts from their internal and communicating actions (by "hiding" them). Furthermore, we also require that some predefined set Γ of (undesired) actions is prevented from being executed by *a fortiori* excluding all sequences in which actions from Γ do occur. Therefore, we "hide" those output actions involved in communications and we define the *observational behaviour* (w.r.t. actions not in Γ) of the resulting team automata as those subsequences of the (remaining) external behaviour that consist solely of actions not in Γ.

Definition 8 *Let $\mathcal{T} = (Q, (\Sigma_{inp}, \Sigma_{out}, \Sigma_{int}), \delta, I)$ be a team automaton over $\mathcal{S}$ and let $\Sigma_{com} \subseteq \Sigma_{ext}$. The* observational behaviour *of $\mathcal{T}$ w.r.t. actions not in $\Gamma \subseteq \Sigma$, denoted by $\mathbf{O}^{\Gamma,\Sigma_{com}}_\mathcal{T}$, is defined as*

$$\mathbf{O}^{\Gamma,\Sigma_{com}}_\mathcal{T} = \mathbf{B}^{\Sigma_{ext}-\Sigma_{com}}_\mathcal{T} \cap (\Sigma - \Gamma)^*.$$

Whenever Γ and Σ_{com} coincide we will simply write $\mathbf{O}^\Gamma_\mathcal{T}$ instead of $\mathbf{O}^{\Gamma,\Gamma}_\mathcal{T}$; note that $\mathbf{O}^\Gamma_\mathcal{T} = \mathbf{B}^{\Sigma_{ext}-\Gamma}_\mathcal{T}$. We are now ready to reformulate (1) in terms of team automata.

Definition 9 *Let $\alpha(\mathcal{T}_P)$ be the expected (correct) behaviour of $\mathcal{T}_P$. Then*

$$\mathcal{T}_P \in GNDC^{\alpha(\mathcal{T}_P),\mathsf{C}}_{\subseteq} \ \textit{ iff } \ \forall\, \mathcal{X} \in \mathcal{E}^\phi_\mathsf{C} : \mathbf{O}^\mathsf{C}_{hide_{\mathsf{C}}(\,|||\,\{\mathcal{T}_P,\mathcal{X}\})} \subseteq \alpha(\mathcal{T}_P).$$

Informally, Definition 9 says that $\mathcal{T}_P$ (*i.e.,* a communication protocol specified in the insecure communication scenario) satisfies $GNDC_{\subseteq}^{\alpha(\mathcal{T}_P),\mathsf{C}}$ if and only if its observational behaviour, despite communicating with any intruder $\mathcal{X}$ through the actions $\mathsf{C} = \Sigma_{com}^{I}$, is included in $\alpha(\mathcal{T}_P)$ (*i.e.,* in the expected correct behaviour of the communication protocol specified by $\mathcal{T}_P$). Significative instances of α will be formalized and used in Section 5 to express integrity and secrecy. Additionally, Definition 9 requires the intruder to be any team automaton able to interact with $\mathcal{T}_P$ through the actions C and with an initial knowledge bound to $\mathsf{D}(\phi)$.

4.3. Two Analysis Strategies for Team Automata

While allowing a uniform approach to specify security properties, Definition 9 does not provide effective strategies for the analysis of security properties of (cryptographic) communication protocols. The universal quantification over $\mathcal{E}_{\mathsf{C}}^{\phi}$ causes serious problems when deciding whether $\mathcal{T}_P \in GNDC_{\subseteq}^{\alpha(\mathcal{T}_P),\mathsf{C}}$. However, the theory developed for GNDC in terms of process algebrae inspires similar methodologies for team automata.

4.3.1. The Most General Intruder

As a first analysis strategy we give a static characterization of the intruder, not involving the universal quantification of Definition 9.

One reasonable way to avoid the infinite number of checks that the universal quantification would require is to study whether there is an attacker that is more powerful (w.r.t. a chosen behavioural relation) than all others. In this way one could reduce the analysis against any environment to an analysis against only one, albeit very powerful, so-called *most general intruder*. Inspired by the theory of GNDC [22], this is exactly what we do. To this aim, we shall make use of a monotonocity property of the team automata operations of composition and hiding. This result guarantees that behavioural inclusion is preserved over these operators:

Lemma 1 *Let $\mathcal{T} = (Q, (\Sigma_{inp}, \Sigma_{out}, \Sigma_{int}), \delta, I)$ be a team automaton and let $\mathcal{X}, \mathcal{X}' \in \mathcal{E}_{\mathsf{C}}$. Then*

$$\mathbf{B}_{\mathcal{X}}^{\mathsf{C}} \subseteq \mathbf{B}_{\mathcal{X}'}^{\mathsf{C}} \text{ implies } \mathbf{O}_{hide_{\mathsf{C}}(|||\{\mathcal{T},\mathcal{X}\})}^{\mathsf{C}} \subseteq \mathbf{O}_{hide_{\mathsf{C}}(|||\{\mathcal{T},\mathcal{X}'\})}^{\mathsf{C}}.$$

Proof. Let $a_1 \cdots a_n \in \mathbf{O}^{\mathsf{C}}_{\mathrm{hide}_{\mathsf{C}}(|||\{\mathcal{T},\mathcal{X}\})}$ and let $\mathbf{B}_{\mathcal{X}}^{\mathsf{C}} \subseteq \mathbf{B}_{\mathcal{X}'}^{\mathsf{C}}$. By (2), the set $\Sigma_{ext}^{\mathcal{X}}$ of external actions of $\mathcal{X}$ is included in C because $\mathcal{X} \in \mathcal{E}_{\mathsf{C}}$. Then, by Definition 8, $a_i \in \Sigma_{ext} - \mathsf{C}$ for all $i \in [n]$. We now use that by definition also all prefixes of $a_1 \cdots a_n$ are included in $\mathbf{O}^{\mathsf{C}}_{\mathrm{hide}_{\mathsf{C}}(|||\{\mathcal{T},\mathcal{X}\})}$ and show by induction that all prefixes of $a_1 \cdots a_n$ are also included in $\mathbf{O}^{\mathsf{C}}_{\mathrm{hide}_{\mathsf{C}}(|||\{\mathcal{T},\mathcal{X}'\})}$. First consider a_1 (the empty behaviour λ is trivial). By Definition 8, either $a_1 \in \mathbf{B}_{\mathrm{hide}_{\mathsf{C}}(|||\{\mathcal{T},\mathcal{X}\})}$ or $b_1 \cdots b_m a_1 \in \mathbf{B}_{\mathrm{hide}_{\mathsf{C}}(|||\{\mathcal{T},\mathcal{X}\})}$ for some $m \geq 1$ and where, for all $j \in [m]$, b_j is an internal action of $\mathrm{hide}_{\mathsf{C}}(|||\{\mathcal{T},\mathcal{X}\})$. In both cases, since $\mathbf{B}_{\mathcal{X}}^{\mathsf{C}} \subseteq \mathbf{B}_{\mathcal{X}'}^{\mathsf{C}}$ and $a_i \in \Sigma_{ext} - \mathsf{C}$ for all $i \in [n]$, it follows by Definition 8 that $a_1 \in \mathbf{O}^{\mathsf{C}}_{\mathrm{hide}_{\mathsf{C}}(|||\{\mathcal{T},\mathcal{X}'\})}$. Now assume that $a_1 \cdots a_k \in \mathbf{O}^{\mathsf{C}}_{\mathrm{hide}_{\mathsf{C}}(|||\{\mathcal{T},\mathcal{X}'\})}$, with $k <$

n, and consider $a_1 \cdots a_{k+1}$. Using the above arguments as above and the induction hypothesis it follows that $a_1 \cdots a_{k+1} \in \mathbf{O}^{\mathsf{C}}_{\mathrm{hide}_{\mathsf{C}}(\,|||\,\{\mathcal{T},\mathcal{X}'\})}$. □

Since $\mathcal{E}^{\phi}_{\mathsf{C}} \subseteq \mathcal{E}_{\mathsf{C}}$, this lemma holds for $\mathcal{X}, \mathcal{X}' \in \mathcal{E}^{\phi}_{\mathsf{C}}$ as well. Based on the approach of [23] we now define a component automaton Top^{ϕ}_{C}, representing the most general intruder mentioned above, in order to circumvent the universal quantification of Definition 9. Recall that the set $\mathsf{C} = \Sigma^{I}_{com}$ of predicates that the intruder uses to interact with the insecure connection is partitioned into C_{inp} and C_{out}, *i.e.,* the predicates of the channels the intruder uses to retrieve messages from and to inject messages back into the insecure connection, resp. (*e.g.,* in Fig. 1, $\mathsf{C}_{inp} = \{\mathsf{Eve}\}$ and $\mathsf{C}_{out} = \{\mathsf{Eve}'\}$).

To ease a comparison with [36], we specify Top^{ϕ}_{C} in the way an I/O automaton is commonly defined [37]: Its states are defined by the current values of the variables listed under States, while its transitions are defined, per action a, as preconditions (Pre) and effect (Eff), *i.e.,* (q, a, q') is a transition of Top^{ϕ}_{C} if the precondition of a is satisfied by q, while q' is the transformation of q defined by the effect of a. We do not specify the precondition or effect of an action when it is true.

Recall that C, C_{inp} and C_{out} are shortcuts for $\mathsf{C}(\mathsf{Messages})$, $\mathsf{C}_{inp}(\mathsf{Messages})$ and $\mathsf{C}_{out}(\mathsf{Messages})$, resp., and that $\mathsf{C} = \mathsf{C}_{inp} \cup \mathsf{C}_{out}$.

Top^{ϕ}_{C}

Actions

Inp: $\mathsf{C}_{inp}(\mathsf{Messages})$ Out: $\mathsf{C}_{out}(\mathsf{Messages})$ Int: $\varnothing$

States

$\mathsf{received} \subseteq \mathsf{Messages}$, initially ϕ

Transitions

$c(m) \in \mathsf{C}_{inp}(\mathsf{Messages})$
Eff: $\mathsf{received} := \mathsf{received} \cup \{m\}$

$c(m) \in \mathsf{C}_{out}(\mathsf{Messages})$
Pre: $m \in \mathsf{D}(\mathsf{received})$

Top^{ϕ}_{C} thus has the possibility to output all messages it receives, possibly after first having applied an inference rule from the cryptosystem to it. It is clear, however, that all sequences consisting of solely locally-controlled actions that Top^{ϕ}_{C} can output (*i.e.,* $\mathbf{B}_{Top^{\phi}_{\mathsf{C}}} \cap (\Sigma^{Top^{\phi}_{\mathsf{C}}}_{loc})^* = \mathbf{B}_{Top^{\phi}_{\mathsf{C}}} \cap \mathsf{C}^*_{out}$) are sequences over $\mathsf{C}(\mathsf{D}(\phi))$: $\mathbf{B}_{Top^{\phi}_{\mathsf{C}}} \cap \mathsf{C}^*_{out} \subseteq (\mathsf{C}(\mathsf{D}(\phi)))^*$. This, together with the fact that we specified $\Sigma^{Top^{\phi}_{\mathsf{C}}}_{inp} = \mathsf{C}_{inp}$ and $\Sigma^{Top^{\phi}_{\mathsf{C}}}_{out} = \mathsf{C}_{out}$, by (2) and (3) implies that $Top^{\phi}_{\mathsf{C}} \in \mathcal{E}^{\phi}_{\mathsf{C}}$. The general way in which Top^{ϕ}_{C} is specified now guarantees that its behaviour includes that of any automaton from $\mathcal{E}^{\phi}_{\mathsf{C}}$:

Lemma 2 *For all* $\mathcal{X} \in \mathcal{E}^{\phi}_{\mathsf{C}}$, $\mathbf{B}^{\mathsf{C}}_{\mathcal{X}} \subseteq \mathbf{B}^{\mathsf{C}}_{Top^{\phi}_{\mathsf{C}}}$.

Proof. Let $\mathcal{X} \in \mathcal{E}^{\phi}_{\mathsf{C}}$. Then (3) implies that $\mathcal{X} \in \mathcal{E}_{\mathsf{C}}$ and thus, by (2) and the specification of Top^{ϕ}_{C}, the input and output actions of $\mathcal{X}$ are included in the input and

output actions of Top^{ϕ}_{C}, resp.. From (3) and the specification of Top^{ϕ}_{C} it now follows immediately that $\mathbf{B}^{\mathsf{C}}_{\mathcal{X}} \subseteq \mathbf{B}^{\mathsf{C}}_{Top^{\phi}_{\mathsf{C}}}$. □

Lemmata 1 and 2 directly imply the following result.

Theorem 2 *For all* $\mathcal{X} \in \mathcal{E}^{\phi}_{\mathsf{C}}$, $\mathbf{O}^{\mathsf{C}}_{hide_{\mathsf{C}}(\,|||\,\{\mathcal{T}_P,\mathcal{X}\})} \subseteq \mathbf{O}^{\mathsf{C}}_{hide_{\mathsf{C}}(\,|||\,\{\mathcal{T}_P,Top^{\phi}_{\mathsf{C}}\})}$.

Together with Definition 9, this gives us the following GNDC schema in terms of team automata.

Corollary 1 *Let* $\alpha(\mathcal{T}_P)$ *be the expected (correct) behaviour of* $\mathcal{T}_P$. *Then*

$$\mathcal{T}_P \in GNDC^{\alpha(\mathcal{T}_P),\mathsf{C}}_{\subseteq} \text{ iff } \mathbf{O}^{\mathsf{C}}_{hide_{\mathsf{C}}(\,|||\,\{\mathcal{T}_P,Top^{\phi}_{\mathsf{C}}\})} \subseteq \alpha(\mathcal{T}_P).$$

By circumventing the universal quantification of Definition 9 we have obtained a static characterization of the intruder, thus easing the use of our GNDC schema in terms of team automata in practice. That it is not always easy (or even feasible) to circumvent the universal quantification in GNDC-like schemata may be concluded from [32], where a Probabilistic Timed NDC schema which contains a universal quantification is introduced.

4.3.2. Compositional Analysis

We now describe a compositional analysis strategy for the insecure communication scenario of Section 3.

To begin with, we fix some notations for the sequel. We let

$$\mathcal{T}_{SIC} = \mathrm{hide}_{\Sigma^P_{com}}(\,|||\,\{\mathcal{T}_S,\mathcal{T}_{IC}\}) \text{ and } \mathcal{T}_{RIC} = \mathrm{hide}_{\Sigma^P_{com}}(\,|||\,\{\mathcal{T}_R,\mathcal{T}_{IC}\})$$

and we let Σ_{SIC} and Σ_{RIC} be their respective sets of actions. Recall that $\mathcal{T}_P$ represents the communication scenario in which an initiator and a responder are connected by an insecure connection, but not yet connected to an intruder. By adding the most general intruder, a general compositional analysis strategy results. Therefore, we let

$$\mathcal{T}_{SX} = \mathrm{hide}_{\mathsf{C}}(\,|||\,\{\mathcal{T}_{SIC}, Top^{\phi}_{\mathsf{C}}\}) \text{ and } \mathcal{T}_{RX} = \mathrm{hide}_{\mathsf{C}}(\,|||\,\{\mathcal{T}_{RIC}, Top^{\phi}_{\mathsf{C}}\})$$

and we let Σ_{SX} and Σ_{RX} be their respective sets of actions.

Lemma 3 *Let* $\{\, m \mid c(m) \in \Sigma^P_{com} \,\} \subseteq \phi$. *Then*

$$\mathbf{O}^{\mathsf{C}}_{hide_{\mathsf{C}}(\,|||\,\{\,|||\,\{\mathcal{T}_{SIC},\mathcal{T}_{RIC}\},Top^{\phi}_{\mathsf{C}}\})} = \underline{||}_{\{\Sigma_{SX},\Sigma_{RX}\}}\ \{\mathbf{O}^{\mathsf{C}}_{\mathcal{T}_{SX}}, \mathbf{O}^{\mathsf{C}}_{\mathcal{T}_{RX}}\}.$$

Proof. From the way that $\mathcal{T}_{SX}$ and $\mathcal{T}_{RX}$ are composed it follows that Σ^S_{sig} and Σ^R_{sig} are the external actions of $\mathcal{T}_{SX}$ and $\mathcal{T}_{RX}$, resp.. Subsequently, let $\mathcal{T}'' = \mathrm{hide}_{\mathsf{C}}(\,|||\,\{\,|||\,\{\mathcal{T}_{SIC},\mathcal{T}_{RIC}\}, Top^{\phi}_{\mathsf{C}}\})$. Then it remains to prove that $\mathbf{O}^{\mathsf{C}}_{\mathcal{T}''} = \underline{||}_{\{\Sigma_{SX},\Sigma_{RX}\}}\ \{\mathbf{O}^{\mathsf{C}}_{\mathcal{T}_{SX}}, \mathbf{O}^{\mathsf{C}}_{\mathcal{T}_{RX}}\}$. Since $\Sigma^S_{sig} \cap \Sigma^R_{sig} = \varnothing$ implies that $\mathbf{O}^{\mathsf{C}}_{\mathcal{T}_{SX}} \cap \mathbf{O}^{\mathsf{C}}_{\mathcal{T}_{RX}} = \varnothing$, this however follows directly from the fact that $\{\, m \mid c(m) \in \Sigma^P_{com} \,\} \subseteq \phi$, *i.e.,* adding $\mathcal{T}_R$ to $\mathcal{T}_{SX}$ in order to obtain $\mathcal{T}''$ does not change the signals from Σ^S_{sig} that $\mathcal{T}_S$ can

output because all messages that $\mathcal{T}_R$ can send to $\mathcal{T}_{IC}$ have already been included in the initial knowledge ϕ of Top_{C}^{ϕ} (and, equivalently, adding $\mathcal{T}_S$ to $\mathcal{T}_{RX}$ does not change the output of $\mathcal{T}_R$). □

Before continuing, we need the following obvious property of full synchronized shuffles.

Remark 2 For $i \in [4]$, let Δ_i be a set of symbols and $L_i \subseteq \Delta_i^*$. Then clearly $\underline{\|}_{\{\Delta_1,\Delta_3\}} \{L_1, L_3\} \subseteq \underline{\|}_{\{\Delta_2,\Delta_4\}} \{L_2, L_4\}$ whenever $L_1 \subseteq L_2$ and $L_3 \subseteq L_4$.

We are now ready to prove our compositional analysis strategy. Whenever $\mathcal{T}_{SIC}$ and $\mathcal{T}_{RIC}$ satisfy *GNDC* w.r.t. their observational behaviour, then the max-ai team automaton $|||\{\mathcal{T}_{SIC}, \mathcal{T}_{RIC}\}$ over $\mathcal{T}_{SIC}$ and $\mathcal{T}_{RIC}$ satisfies *GNDC* w.r.t. the full synchronized shuffle of their observational behaviours:

Theorem 3 *Let* $\{\, m \mid c(m) \in \Sigma_{com}^P \,\} \subseteq \phi$. *Then*

$$\mathcal{T}_{SIC} \in GNDC_{\subseteq}^{\mathbf{O}_{\mathcal{T}_{SIC}}^{\mathsf{C}},\mathsf{C}} \text{ and } \mathcal{T}_{RIC} \in GNDC_{\subseteq}^{\mathbf{O}_{\mathcal{T}_{RIC}}^{\mathsf{C}},\mathsf{C}} \text{ implies } |||\{\mathcal{T}_{SIC}, \mathcal{T}_{RIC}\} \in GNDC_{\subseteq}^{\underline{\|}_{\{\Sigma_{SIC},\Sigma_{RIC}\}} \{\mathbf{O}_{\mathcal{T}_{SIC}}^{\mathsf{C}},\mathbf{O}_{\mathcal{T}_{RIC}}^{\mathsf{C}}\},\mathsf{C}}.$$

Proof. Let $\mathcal{T}_{SIC} \in GNDC_{\subseteq}^{\mathbf{O}_{\mathcal{T}_{SIC}}^{\mathsf{C}},\mathsf{C}}$ and $\mathcal{T}_{RIC} \in GNDC_{\subseteq}^{\mathbf{O}_{\mathcal{T}_{RIC}}^{\mathsf{C}},\mathsf{C}}$. Then, by Corollary 1, $\mathbf{O}_{\mathcal{T}_{SX}}^{\mathsf{C}} \subseteq \mathbf{O}_{\mathcal{T}_{SIC}}^{\mathsf{C}}$ and $\mathbf{O}_{\mathcal{T}_{RX}}^{\mathsf{C}} \subseteq \mathbf{O}_{\mathcal{T}_{RIC}}^{\mathsf{C}}$ and thus, by Lemma 3 and Remark 2, we have $\mathbf{O}_{\mathrm{hide}_{\mathsf{C}}(|||\{|||\{\mathcal{T}_{SIC},\mathcal{T}_{RIC}\},Top_{\mathsf{C}}^{\phi}\})}^{\mathsf{C}} = \underline{\|}_{\{\Sigma_{SX},\Sigma_{RX}\}} \{\mathbf{O}_{\mathcal{T}_{SX}}^{\mathsf{C}}, \mathbf{O}_{\mathcal{T}_{RX}}^{\mathsf{C}}\} \subseteq \underline{\|}_{\{\Sigma_{SIC},\Sigma_{RIC}\}} \{\mathbf{O}_{\mathcal{T}_{SIC}}^{\mathsf{C}}, \mathbf{O}_{\mathcal{T}_{RIC}}^{\mathsf{C}}\}$. □

We have thus defined a first compositional analysis strategy for our insecure communication scenario for team automata, considering one initiator (sender) and one responder (receiver). It remains to be seen how to extend this result to the case of multiple receivers and thus extend its applicability to the analysis of multi-party protocols. In the next section we will apply this compositional analysis strategy in a case study to verify integrity.

5. Two Case Studies

In this section we apply the framework developed in the previous sections in two different case studies. In Section 5.1 we use the GNDC schema in terms of team automata, together with the insecure communication scenario for team automata, to show that integrity in the sense of robustness of modifications is guaranteed in the so-called deterministic (1,2) schema of the EMSS protocol. In Section 5.2, on the other hand, we show how secrecy is preserved when a member of a multicast group leaves the group in a particular run of the complementary variable approach to the N-Root/Leaf pairwise keys protocol. The first case study is taken from [6], while the second case study has not been presented before.

We repeat that the aim of the two case studies is not in the first place to provide new insights into the involved protocols, but rather to illustrate our approach for two

well-known protocols, thus allowing an easy comparison for those familiar with other approaches. In fact, integrity for the (1,2) schema of the EMSS protocol has already been validated in [38], where a CCS-like process algebra was used instead.

5.1. The EMSS Protocol: Proving Integrity

The EMSS protocol was introduced in [42] and is used to sign digital streams. It exploits a combination of hash functions and digital signatures and it achieves some robustness against packet loss, *i.e.,* an incompletely received stream still allows the user to verify the integrity of the packets that were not lost.

Actually EMSS is a family of protocols and here we focus on its deterministic (1,2) schema. We assume that a sender S wants to send a stream of payloads $m_0, m_1, \ldots, m_{last}$ to a set of receivers $\{ R_n \mid n \geq 1 \}$ (as is usual for recipients of digital data streams, we assume that the receivers are not able to communicate among each other and that the private sender key $sk(S)$ cannot be deduced from $\{ m_i \mid 0 \leq i \leq last \}$). The protocol then requires S to send triples (called packets) built from payloads to the receivers. After an initial phase, each packet P_i contains a meaningful payload m_i, together with the hashes $h(P_{i-1})$ and $h(P_{i-2})$ of the previous two packets sent. The end of a stream is indicated by a signature packet P_{sign} containing the hashes of the final two packets, along with a digital signature of those hashes. In this way, some robustness against packet loss is achieved. For the sake of readability, we only represent the signature in packet P_{sign}.

The deterministic (1,2) schema of the EMSS protocol can be formally described as:

$$
\begin{array}{lllll}
S \xrightarrow{P_0} \{ R_n \mid n \geq 1 \} & \text{packet} & P_0 & = \langle m_0, \varnothing, \varnothing \rangle & \\
S \xrightarrow{P_1} \{ R_n \mid n \geq 1 \} & \text{packet} & P_1 & = \langle m_1, h(P_0), \varnothing \rangle & \\
S \xrightarrow{P_i} \{ R_n \mid n \geq 1 \} & \text{packet} & P_i & = \langle m_i, h(P_{i-1}), h(P_{i-2}) \rangle & 2 \leq i \leq last \\
S \xrightarrow{P_{sign}} \{ R_n \mid n \geq 1 \} & \text{packet} & P_{sign} & = \langle \{ h(P_{last}), h(P_{last-1}) \}_{sk(S)} \rangle &
\end{array}
$$

5.1.1. Modelling the EMSS Protocol with Team Automata

The sender S is modelled by a component automaton $\mathcal{T}_S$ and the set $\{ R_n \mid n \geq 1 \}$ of receivers by n copies $\mathcal{T}_R^{(1)}, \ldots, \mathcal{T}_R^{(n)}$ of a component automaton $\mathcal{T}_R$. The internal actions of each $\mathcal{T}_R^{(i)}$ are assumed to be indexed in order to satisfy composability. Our analysis assumes one $\mathcal{T}_R$, but since multiple receivers cannot communicate among each other, this is not a real restriction. $\mathcal{T}_S$ uses its private key $sk(\mathcal{T}_S)$ and a public key $pk(\mathcal{T}_S)$ to perform digital signature operations. $\mathsf{Payloads}$ denotes the set $\{m_0, m_1, \ldots, m_{last}\}$ of meaningful payloads, with $0 \leq i \leq last$. $\mathcal{T}_S$ uses the hash function $h : \mathsf{Packets} \to \mathsf{Hashed}$, while $\mathcal{T}_R$ uses the hash function $\bar{h} : \mathsf{Packets} \to \mathsf{Hashed}$. Moreover, $\mathcal{T}_S$ uses the function $s : 2^{\mathsf{Hashed}} \to \mathsf{Signed}$, defined by $s(H) = H_{sk(\mathcal{T}_S)}$, to sign sets of hashed packets with its private key $sk(\mathcal{T}_S)$, whereas $\mathcal{T}_R$ uses the function $\bar{s} : \mathsf{Signed} \to \{\mathsf{true}, \mathsf{false}\}$ and the public key $pk(\mathcal{T}_S)$ to verify whether a set of hashed packets was signed by $\mathcal{T}_S$.

The full specification of the sender component automaton $\mathcal{T}_S$ is as follows.

$\mathcal{T}_S$

Actions

Inp: $\varnothing$

Out: $\{\overbrace{\langle m_0, \varnothing, \varnothing\rangle}^{P_0}, \overbrace{\langle m_1, h(P_0), \varnothing\rangle}^{P_1}\} \cup \{\overbrace{\langle m_i, h(P_{i-1}), h(P_{i-2})\rangle}^{P_i} \mid 2 \le i \le last\} \cup \{\underbrace{\langle\{h(P_{last}), h(P_{last-1})\}_{sk(\mathcal{T}_S)}\rangle}_{P_{sign}}\}$

Int: $\{Hash_i \mid 0 \le i \le last\} \cup \{Sign\}$

States

$\mathsf{sent} \subseteq \mathsf{Messages}$, $\mathsf{hashed} \subseteq \mathsf{Hashed}$, $\mathsf{signed} \subseteq \mathsf{Signed}$, all initially $\varnothing$

Transitions

P_0
- Pre: $P_0 \notin \mathsf{sent}$
- Eff: $\mathsf{sent} := \mathsf{sent} \cup \{P_0\}$

$Hash_i,\ 0 \le i \le last$
- Pre: $P_i \in \mathsf{sent} \wedge h(P_i) \notin \mathsf{hashed}$
- Eff: $\mathsf{hashed} := \mathsf{hashed} \cup \{h(P_i)\}$

$Sign$
- Pre: $h(P_{last}) \in \mathsf{hashed} \wedge s(\{h(P_{last}), h(P_{last-1})\}) \notin \mathsf{signed}$
- Eff: $\mathsf{signed} := \mathsf{signed} \cup \{s(\{h(P_{last}), h(P_{last-1})\})\}$

P_1
- Pre: $h(P_0) \in \mathsf{hashed} \wedge P_1 \notin \mathsf{sent}$
- Eff: $\mathsf{sent} := \mathsf{sent} \cup \{P_1\}$

$P_i,\ 2 \le i \le last$
- Pre: $\{h(P_{i-1}), h(P_{i-2})\} \subseteq \mathsf{hashed} \wedge P_i \notin \mathsf{sent}$
- Eff: $\mathsf{sent} := \mathsf{sent} \cup \{P_i\}$

P_{sign}
- Pre: $\{h(P_{last}), h(P_{last-1})\}_{sk(\mathcal{T}_S)} \in \mathsf{signed} \wedge P_{sign} \notin \mathsf{sent}$
- Eff: $\mathsf{sent} := \mathsf{sent} \cup \{P_{sign}\}$

Clearly $\mathcal{T}_S$ has no input behaviour, while its output behaviour $\mathbf{B}_{\mathcal{T}_S}^{\Sigma_{out}}$ consists of all prefixes of $P_0P_1 \cdots P_{last}P_{sign}$. To send the packets $P_0, P_1, \ldots, P_{last}, P_{sign}$ in this order, $\mathcal{T}_S$ must perform some internal computations. This is reflected by its internal behaviour $\mathbf{B}_{\mathcal{T}_S}^{\Sigma_{int}}$ consisting of all prefixes of $Hash_0 Hash_1 \cdots Hash_{last} Sign$.

We continue with the specification of the receiver $\mathcal{T}_R$. It is capable of receiving as input behaviour all packets $P_0, P_1, \ldots, P_{last}, P_{sign}$, built over the set $\mathsf{Payloads}'$ of variables m_i' that should contain the meaningful payloads m_i. Upon receiving P_i, $\mathcal{T}_R$ verifies whether it has received P_{i-1}. First consider $\mathcal{T}_R$ indeed received P_{i-1}. Then it extracts the hash $h(P_{i-1})$ from P_i, computes the hash $\bar{h}(P_{i-1})$, and compares these two hashes. If they are equal, then the variable m_{i-1}' that should contain the verifiable payload m_{i-1} is extracted from P_{i-1}. Otherwise $\mathcal{T}_R$ has no output behaviour.

Secondly, consider $\mathcal{T}_R$ did not receive P_{i-1}. Then it verifies whether it received P_{i-2}. If it did not, then $\mathcal{T}_R$ concludes it is unable to check the hashes of either P_{i-1} or P_{i-2}, so it goes on to verify whether it did receive P_{i+1}. If $\mathcal{T}_R$ did receive P_{i-2}, then it extracts the hash $h(P_{i-2})$ from P_i, computes the hash $\bar{h}(P_{i-2})$, and compares the two hashes. If they are equal, then the variable m_{i-2}' that should contain the verifiable payload m_{i-2} is extracted from P_{i-2}. Otherwise $\mathcal{T}_R$ has no output behaviour.

Eventually $\mathcal{T}_R$ receives the signature packet P_{sign} (we assume that P_{sign} is always received, but in the specification of $\mathcal{T}_R$ we do sometimes check whether P_{sign} has

already been received to avoid a transition to occur *before* P_{sign} was indeed received), after which it verifies the accompanying digital signature (we assume that $\mathcal{T}_R$ has previously retrieved the public key $pk(\mathcal{T}_S)$ corresponding to the private key $sk(\mathcal{T}_S)$), before repeating the above procedure. The verification of the signature allows $\mathcal{T}_R$ to have guarantees on the integrity of the stream of verifiable payloads collected in xtractedM, which is then sent to the application level as the output behaviour of $\mathcal{T}_R$.

Note that in the specification of $\mathcal{T}_S$ we explicitly modelled that each of its actions is enabled only once during a computation, thus prohibiting loops. For example, as soon as $\mathcal{T}_S$ has sent P_0, then this action's precondition $P_0 \notin \mathsf{sent}$ prohibits this action to be executed again. For the sake of readability, we omit the addition of such preconditions to the following full specification of the receiver component automaton $\mathcal{T}_R$, but we implicitly assume that its actions are executed only once during a computation.

$\mathcal{T}_R$

Actions

Inp: $\{\overbrace{\langle m'_0, \varnothing, \varnothing\rangle}^{P_0}, \overbrace{\langle m'_1, h(P_0), \varnothing\rangle}^{P_1}\} \cup \{\overbrace{\langle m'_i, h(P_{i-1}), h(P_{i-2})\rangle}^{P_i} \mid 2 \le i \le last\} \cup \{\underbrace{\langle\{h(P_{last}), h(P_{last-1})\}_{sk(\mathcal{T}_S)}\rangle}_{P_{sign}}\}$

Out: $\mathsf{Payloads}'$

Int: $\{XtractH_{i,j}, XtractM_i, Hash_i \mid 0 \le i \le last, j = 1, 2\} \cup \{XtractH_{sign}, Stream\}$

States

$\mathsf{received}, \mathsf{xtractedM} \subseteq \mathsf{Payloads}'$, $\mathsf{xtractedH}, \mathsf{hashed} \subseteq \mathsf{Hashed}$, all initially $\varnothing$

$\mathsf{verified}_0, \mathsf{verified}_1, \ldots, \mathsf{verified}_{last}, \mathsf{verified}_{sign}, \mathsf{send} \subseteq \{\mathsf{true}, \mathsf{false}\}$, all initially false

Transitions

$P_i,\ 0 \le i \le last$ or $i = sign$
Pre: $P_{sign} \notin \mathsf{received}$
Eff: $\mathsf{received} := \mathsf{received} \cup \{P_i\}$

$XtractH_{sign}$
Pre: $P_{sign} \in \mathsf{received} \wedge \bar{s}(\{h(P_{last}), h(P_{last-1})\}_{sk(\mathcal{T}_S)}) = \mathsf{true}$
Eff: $\mathsf{verified}_{sign} := \mathsf{true}$, $\mathsf{xtractedH} := \mathsf{xtractedH} \cup \{h(P_{last-1}), h(P_{last})\}$

$Hash_i,\ 0 \le i \le last$
Pre: $h(P_i) \in \mathsf{xtractedH} \wedge P'_i \in \mathsf{received}$
Eff: $\mathsf{hashed} := \mathsf{hashed} \cup \{\bar{h}(P_i)\}$

$XtractM_i,\ 0 \le i \le last$
Pre: $\bar{h}(P_i) \in \mathsf{hashed} \wedge \bar{h}(P_i) = h(P_i)$
Eff: $\mathsf{verified}_i := \mathsf{true}$, $\mathsf{xtractedM} := \mathsf{xtractedM} \cup \{m'_i\}$

$XtractH_{i,1},\ 1 \le i \le last$
Pre: $\mathsf{verified}_i = \mathsf{true} \wedge \{P_{i-1}, P_i\} \subseteq \mathsf{received}$
Eff: $\mathsf{xtractedH} := \mathsf{xtractedH} \cup \{h(P_{i-1})\}$

$XtractH_{i,2},\ 2 \le i \le last$
Pre: $\mathsf{verified}_i = \mathsf{true} \wedge \{P_{i-2}, P_i\} \subseteq \mathsf{received} \wedge P_{i-1} \notin \mathsf{received}$
Eff: $\mathsf{xtractedH} := \mathsf{xtractedH} \cup \{h(P_{i-2})\}$

$Stream$
Pre: $\mathsf{verified}_{sign} = \mathsf{true} \wedge \forall\, 0 \le i \le last : P_i \in \mathsf{received} \Rightarrow \mathsf{verified}_i = \mathsf{true}$
Eff: $\mathsf{send} := \mathsf{true}$

$m'_i,\ 0 \le i \le last$
Pre: $\mathsf{send} = \mathsf{true} \wedge m'_i \in \mathsf{xtractedM} \wedge \{m'_k \mid 0 \le k < i\} \cap \mathsf{xtractedM} = \varnothing$
Eff: $\mathsf{xtractedM} := \mathsf{xtractedM} - \{m'_i\}$

Clearly the input behaviour $\mathbf{B}_{\mathcal{T}_R}^{\Sigma_{inp}}$ of $\mathcal{T}_R$ as a stand-alone component automaton consists of all prefixes of all possible permutations of $P_0 P_1 \cdots P_{last} P_{sign}$

that end with P_{sign} (due to the precondition of the P_i's). Whenever $\mathcal{T}_R$ actually receives the packets $P_0, P_1, \ldots, P_{last}, P_{sign}$ in this particular order and in the ideal situation that no packets get lost, then it is able to perform a series of internal computations, reflected by the fact that its internal behaviour $\mathbf{B}_{\mathcal{T}_R}^{\Sigma_{int}}$ contains $XtractH_{sign}Hash_{last}XtractM_{last}Hash_{last-1}XtractM_{last-1}\cdots XtractH_{1,1}Hash_0XtractM_0Stream$ as well as other sequences representing other orders of performing these internal computations that do end with *Stream* however. In this ideal situation, the output behaviour $\mathbf{B}_{\mathcal{T}_R}^{\Sigma_{out}}$ of $\mathcal{T}_R$ consists of all prefixes of $m'_0m'_1\cdots m'_{last}$. Note that the precondition of *Stream* guarantees that the output behaviour always has this particular order. Note moreover that $\mathcal{T}_R$ is constructed such that it can handle a certain degree of packet loss, in which case its behaviour of course slightly changes.

Now the max-ai team automaton over $\{\mathcal{T}_S, \mathcal{T}_R^{(i)} \mid 1 \leq i \leq n\}$, denoted by $\mathcal{T}_{EMSS}$, is defined as

$$\mathcal{T}_{EMSS} = |||\,\{\mathcal{T}_S, \mathcal{T}_R^{(i)} \mid 1 \leq i \leq n\},$$

which formalizes the EMSS protocol. Note that $\mathcal{T}_{EMSS}$ has no input actions, while it has the union of the output (internal, resp.) actions of $\mathcal{T}_S$ and those of $\mathcal{T}_R$ as its output (internal, resp.) actions. The fact that its output behaviour consists of all prefixes of $P_0P_1\cdots P_{last}P_{sign}m'_0m'_1\cdots m'_{last}$ implies that it models broadcast communication (the definition of the max-ai team automaton guarantees that all $\mathcal{T}_R$'s participate in each action, *i.e.,* receive all packets and output all messages). By composing other team automata, we can model multicast communication and even packet loss. The more detailed description of this case study in [6] deals with such issues.

5.1.2. Analyzing the EMSS Protocol with Team Automata

We define integrity as the ability of $\mathcal{T}_R$ to accept a message m_i, for any i, only as the ith message sent by $\mathcal{T}_S$. We moreover assume that $\mathcal{T}_R$ signals the acceptance of a stream of messages as a legitimate one by issuing it as a list of messages through special actions $\{\texttt{Reveal}'\}$. We assume the expected (correct) observational behaviour $\alpha_{int}(\mathcal{T}_P)$ of $\mathcal{T}_P$ w.r.t. integrity to be $\alpha_{int}(\mathcal{T}_P) = \mathbf{O}_{\mathcal{T}_P}^{\mathsf{C}}$, *i.e.,* all prefixes of the sequence $\texttt{Reveal}'(m'_0)\texttt{Reveal}'(m'_1)\cdots\texttt{Reveal}'(m'_{last})$. Note that following the notation introduced at the beginning of Section 3, actions may henceforth be written as composed terms, *e.g.,* action $\texttt{Reveal}'(m'_0)$ consists of an outermost part $\texttt{Reveal}'$ that is a predicate symbol and an innermost part m'_0 that is an atomic formula.

Consequently we equip Top_{C}^{ϕ} with an initial knowledge ϕ consisting of all output actions of $\mathcal{T}_S$ and the public key $pk(\mathcal{T}_S)$, *i.e.,* $\phi = \{\, P_0, P_1, P_i, P_{sign} \mid 2 \leq i \leq last \,\} \cup \{pk(\mathcal{T}_S)\}$, where $P_0 = \langle m_0, \varnothing, \varnothing\rangle$, $P_1 = \langle m_1, h(P_0), \varnothing\rangle$, $P_i = \langle m_i, h(P_{i-1}), h(P_{i-2})\rangle$, for all $2 \leq i \leq last$, and $P_{sign} = \langle\{h(P_{last}), h(P_{last-1})\}_{sk(\mathcal{T}_S)}\rangle$. Without loss of generality we do so for purely technical reasons, *viz.* to enable Top_{C}^{ϕ} to send the correct messages to $\mathcal{T}_R$ over the insecure connection when analyzing the max-ai team automaton over $\mathcal{T}_{RIC}$ and Top_{C}^{ϕ} (*cf.* the forthcoming proof of Theorem 5). In fact, the messages contained in its initial knowledge are exactly those that it is anyway able to receive in the max-ai team automaton over $\mathcal{T}_P$ and Top_{C}^{ϕ} by eavesdropping when $\mathcal{T}_S$

sends them over the insecure connection. Moreover, as is common for the analysis of (cryptographic) communication protocols, we rely on the *perfect encryption assumption*, *i.e.,* Top^{ϕ}_{C} cannot deduce $sk(\mathcal{T}_S)$ from ϕ nor can it forge hash and encryption functions by guessing.

From Section 4.3.2 we recall that $\mathcal{T}_{SIC} = \mathrm{hide}_{\Sigma^P_{com}}(\,|||\,\{\mathcal{T}_S, \mathcal{T}_{IC}\})$ with actions Σ_{SIC}, $\mathcal{T}_{RIC} = \mathrm{hide}_{\Sigma^P_{com}}(\,|||\,\{\mathcal{T}_R, \mathcal{T}_{IC}\})$ with actions Σ_{RIC}, $\mathcal{T}_{SX} = \mathrm{hide}_{\mathsf{C}}(\,|||\,\{\mathcal{T}_{SIC}, Top^{\phi}_{\mathsf{C}}\})$ with actions Σ_{SX} and $\mathcal{T}_{RX} = \mathrm{hide}_{\mathsf{C}}(\,|||\,\{\mathcal{T}_{RIC}, Top^{\phi}_{\mathsf{C}}\})$ with actions Σ_{RX}.

To begin with, it is clear that the observational behaviour of the max-ai team automaton over $\mathcal{T}_{SIC}$ and Top^{ϕ}_{C} is empty, *i.e.,*

Theorem 4 $\mathcal{T}_{SIC} \in GNDC^{\varnothing,\mathsf{C}}_{\subseteq}$.

Proof. Directly by Corollary 1 because $\mathbf{O}^{\mathsf{C}}_{\mathrm{hide}_{\mathsf{C}}(\,|||\,\{\mathcal{T}_{SIC}, Top^{\phi}_{\mathsf{C}}\})} = \varnothing$. □

We now show that the way the receiver verifies the messages it receives implies that the observational behaviour of the max-ai team automaton over $\mathcal{T}_{RIC}$ and Top^{ϕ}_{C} is included in the expected observational behaviour $\alpha_{int}(\mathcal{T}_P)$ of $\mathcal{T}_P$ w.r.t. integrity, *i.e.,*

Theorem 5 $\mathcal{T}_{RIC} \in GNDC^{\alpha_{int}(\mathcal{T}_P),\mathsf{C}}_{\subseteq}$.

Proof. According to Corollary 1 we need to show that $\mathbf{O}^{\mathsf{C}}_{\mathrm{hide}_{\mathsf{C}}(\,|||\,\{\mathcal{T}_{RIC}, Top^{\phi}_{\mathsf{C}}\})} \subseteq \alpha_{int}(\mathcal{T}_P)$. We distinguish two cases. If $\mathcal{T}_R$ does not output any action m'_i, then its empty observational behaviour is trivially included in $\alpha_{int}(\mathcal{T}_P)$. Next assume that $\mathcal{T}_R$ does output a nonempty sequence of actions m'_i. Then the preconditions of the actions m'_i guarantee that $\mathcal{T}_R$ must output a sequence of the form $m'_{i_0} m'_{i_1} \cdots m'_{i_{last}}$, with $0 \leq i_0 \leq i_1 \leq \cdots \leq i_{last} \leq last$. Without loss of generality, we assume that $\mathcal{T}_R$ outputs the sequence $m'_0 m'_1 \cdots m'_{last}$.

The preconditions of the actions m'_i imply that $\mathsf{send} = \mathsf{true}$ must hold whenever $\mathcal{T}_R$ outputs an action m'_i. The only action by which send can be set to true is *Stream*, which in its turn implies that the precondition of *Stream* must have been satisfied, *i.e.,* for all $0 \leq i \leq last$: If $\mathcal{T}_R$ has received P_i, then $\mathsf{verified}_i = \mathsf{true}$ and, moreover, $\mathsf{verified}_{sign} = \mathsf{true}$. The latter implies that $\mathcal{T}_R$ must have verified that P_{sign} was signed with $sk(\mathcal{T}_S)$. Because Top^{ϕ}_{C} cannot deduce this private key from its initial knowledge and none of the automata ever outputs this private key, it follows that Top^{ϕ}_{C} does not know $sk(\mathcal{T}_S)$ and hence $\mathcal{T}_R$ has received the original packet P_{sign} of $\mathcal{T}_S$. Furthermore, the fact that $\mathcal{T}_R$ has output $m'_0 m'_1 \cdots m'_{last}$ implies that it must have extracted all m'_i's, received all P_i's, and set all $\mathsf{verified}_i$'s to true, which means that $\bar{h}(P_i) = h(P_i)$ must have held for all $0 \leq i \leq last$. Since digital signatures and hash functions cannot be forged by the intruder, the only possibility for $\mathcal{T}_R$ to output $m'_0 m'_1 \cdots m'_{last}$ is that it has received all the original packets of $\mathcal{T}_S$. This shows why, in absence of $\mathcal{T}_S$, we had to equip Top^{ϕ}_{C} with an initial knowledge consisting of all output actions of $\mathcal{T}_S$ (which, once again, it would anyway have received from $\mathcal{T}_S$). Since the behaviour of any team automaton is prefix closed, we conclude that $\mathbf{O}^{\mathsf{C}}_{\mathrm{hide}_{\mathsf{C}}(\,|||\,\{\mathcal{T}_{RIC}, Top^{\phi}_{\mathsf{C}}\})} \subseteq \alpha_{int}(\mathcal{T}_P)$. □

Finally, after an observation on the composition of team automata that have no internal actions, we can show that integrity is guaranteed in the instance of the EMSS protocol under scrutiny.

Remark 3 If $\{\mathcal{T},\mathcal{T}\}$ is a composable system, then clearly $\mathbf{B}_{|||\,\{\mathcal{T},\mathcal{T}\}} = \mathbf{B}_{\mathcal{T}}$.

Theorem 6 $\mathcal{T}_P \in GNDC_{\subseteq}^{\alpha_{int}(\mathcal{T}_P),\mathsf{C}}$.

Proof. By Theorems 3–5, $|||\,\{\mathcal{T}_{SIC},\mathcal{T}_{RIC}\} \in GNDC_{\subseteq}^{||\,\{\Sigma_{SIC},\Sigma_{RIC}\}\ \{\mathbf{O}^{\mathsf{C}}_{\mathcal{T}_{SIC}},\mathbf{O}^{\mathsf{C}}_{\mathcal{T}_{RIC}}\},\mathsf{C}} = GNDC_{\subseteq}^{||\,\{\mathtt{Reveal'}\}\ \{\varnothing,\alpha_{int}(\mathcal{T}_P)\},\mathsf{C}} = GNDC_{\subseteq}^{\alpha_{int}(\mathcal{T}_P),\mathsf{C}}$. Consequently, we obtain that $\mathbf{O}^{\mathsf{C}}_{\mathrm{hide}_{\mathsf{C}}(|||\,\{|||\,\{\mathcal{T}_{SIC},\mathcal{T}_{RIC}\},Top^{\phi}_{\mathsf{C}}\})} \subseteq \alpha_{int}(\mathcal{T}_P)$ by Corollary 1. Since $\mathcal{T}_{IC}$ has no internal actions, $\{\mathcal{T}_{IC},\mathcal{T}_{IC}\}$ forms a composable system. It then follows from Remarks 1 and 3 that $\mathbf{B}_{|||\,\{|||\,\{\mathcal{T}_{SIC},\mathcal{T}_{RIC}\},Top^{\phi}_{\mathsf{C}}\}} = \mathbf{B}_{|||\,\{\mathcal{T}_S,\mathcal{T}_R,\mathcal{T}_{IC},Top^{\phi}_{\mathsf{C}}\}} = \mathbf{B}_{|||\,\{\mathcal{T}_P,Top^{\phi}_{\mathsf{C}}\}}$ and hence $\mathbf{O}^{\mathsf{C}}_{\mathrm{hide}_{\mathsf{C}}(|||\,\{|||\,\{\mathcal{T}_{SIC},\mathcal{T}_{RIC}\},Top^{\phi}_{\mathsf{C}}\})} = \mathbf{O}^{\mathsf{C}}_{\mathrm{hide}_{\mathsf{C}}(|||\,\{\mathcal{T}_P,Top^{\phi}_{\mathsf{C}}\})}$ by Definition 8. This implies that $\mathbf{O}^{\mathsf{C}}_{\mathrm{hide}_{\mathsf{C}}(|||\,\{\mathcal{T}_P,Top^{\phi}_{\mathsf{C}}\})} \subseteq \alpha_{int}(\mathcal{T}_P)$ and thus, by Corollary 1, $\mathcal{T}_P \in GNDC_{\subseteq}^{\alpha_{int}(\mathcal{T}_P),\mathsf{C}}$. □

Note that we have verified integrity in the EMSS protocol in the ideal situation, *i.e.,* without packet loss. By using the more detailed description of this case study in [6] it is straightforward to perform an analysis that does deal with packet loss.

5.2. The Complementary Variable Approach to the N-Root/Leaf Pairwise Keys Protocol: Proving Secrecy

Striving for secrecy within a multicast group means that all and *only* the group members must be able to decrypt transmitted data [11]. Secrecy can be guaranteed by letting the group members share a common group key. To achieve secrecy, the approach presented in [44] is a "brute force method to provide a common multicast group key to the group participants".

The N-Root/Leaf pairwise keys protocol assumes the existence of a multicast session with an initiator that controls the multicast group. This initiator is called the *root* of the group, while each of the remaining N members of the multicast group is called a *leaf*. Here we let S denote the root and we let $\{\,R_i \mid i \in [N]\,\}$ denote the N leaves. In a preliminary phase the root generates a key for each leaf in the multicast group, which the root shares with them by running some standard public-key exchange technique (see, *e.g.,* [35]). The root then generates the group key K and, in order to distribute it to the leaves, it encrypts K with the pairwise keys shared with them. We let K_{SR_i} denote the resulting pairwise key shared between the root S and the leaf R_i, $i \in [N]$.

Since a multicast group may dynamically change, the secrecy of communication within the group should be preserved when a member leaves the group: The parting member should not be able to listen to further communication after its departure.

To this aim, [44] presents the so-called *complementary variable approach* to the N-Root/Leaf pairwise keys protocol: While preparing the first packet P_1, the root S uses a set $V = \{v_1, \ldots, v_N\}$ of complementary variables to guarantee secrecy also in case of membership changes. Along with K, each leaf R_i, $i \in [N]$, receives the personalized set $v_{\setminus i} = V - \{v_i\}$ of complementary variables. Each leaf thus receives a different set of complementary variables, *viz.* leaf R_i receives all variables V except v_i. The root distributes K and these complementary variables by means of a multicast communication of packet $P_1 = \langle \{K, v_{\setminus 1}\}_{K_{SR_1}}, \ldots, \{K, v_{\setminus N}\}_{K_{SR_N}} \rangle$ to the whole group. Upon receiving P_1, each leaf retrieves K from the appropriate segment of the message by using its own secret key it shares with the root. Once K has been distributed, it can be used to multicast encrypted messages to the group. The root S could, *e.g.,* encrypt a message M with K and send it to the group as packet $P_2 = \langle \{M\}_K \rangle$.

Now suppose that leaf R_N leaves the multicast group (it unsubscribes itself). In order to cut that leaf from the group, the root S sends the group a packet $P_3 = \langle cut\ R_N\ off \rangle$ containing a self-explanatory plaintext message. Upon receiving this message in packet P_3, all other leaves generate a new group key K', starting from the old group key K and the complementary variable v_N (*i.e.,* $K' = f(K, v_N)$, where f is an appropriate function). Any further communication within the group will be encrypted with the new group key K'. Leaf R_1 could, *e.g.,* encrypt a message M' with K' and send it to the group as packet $P_4 = \langle \{M'\}_{K'} \rangle$. Since everyone except leaf R_N knows v_N, everyone except leaf R_N is able to generate K'. Secrecy is then intended as the secrecy of M' w.r.t. R_N. In general, in this particular protocol secrecy is thus intended as the secrecy of the messages sent *after* some leaf has left the group, w.r.t. that very leaf.

The above run of the complementary variable approach to the N-Root/Leaf pairwise keys protocol, in which we thus assume that S sends the first message encrypted with K (packet P_2) to the group, while R_1 sends the first message encrypted with K' (packet P_4) to the group, can be formally described as follows.

$$
\begin{array}{lll}
S \xrightarrow{P_1} \{ R_i \mid 1 \leq i \leq N \} & \text{packet } P_1 = \langle \{K, v_{\setminus 1}\}_{K_{SR_1}}, \ldots, \{K, v_{\setminus N}\}_{K_{SR_N}} \rangle \\
S \xrightarrow{P_2} \{ R_i \mid 1 \leq i \leq N \} & \text{packet } P_2 = \langle \{M\}_K \rangle \\
S \xrightarrow{P_3} \{ R_i \mid 1 \leq i \leq N \} & \text{packet } P_3 = \langle cut\ R_N\ off \rangle \\
R_1 \xrightarrow{P_4} \{ R_j \mid 1 < j \leq N \} \cup \{S\} & \text{packet } P_4 = \langle \{M'\}_{K'} \rangle
\end{array}
$$

5.2.1. Modelling the Complementary Variable Approach to the N-Root/Leaf Pairwise Keys Protocol with Team Automata

We now show a partial team automata specification of the complementary variable approach to the N-Root/Leaf pairwise keys protocol. Since our aim is to analyze the secrecy of a message after a member has left the multicast group, the model specifies only those features significant for such an analysis.

We focus on the particular run of the protocol discussed in the previous section. Without loss of generality, we assume the following. The root S uses the group key K to broadcast a particular message M to the group members. Leaf R_N is the member

that leaves the group. R_1 is the group member that multicasts a new particular message M' to the remaining group members. In this scenario, secrecy is preserved if R_N is unable to decrypt M' after its departure.

In the remainder of this section we model root S and leaf R_1 by component automata $\mathcal{T}_S$ and $\mathcal{T}_{R_1}$, resp., and we model the remaining leaves (including leaf R_N that will leave the multicast group) as component automata $\mathcal{T}_{R_i}$, $2 \leq i \leq N-1$.

Let M, M' range over a set Msgs of messages and let K, K' range over a set Keys of group keys.

$\mathcal{T}_S$: the root member

Actions

Inp: $\{\overbrace{\langle\{M'\}_{K'}\rangle}^{P_4}\}$ Out: $\{\overbrace{\langle\{K, v_{\setminus 1}\}_{K_{SR_1}}, \ldots, \{K, v_{\setminus N}\}_{K_{SR_N}}\rangle}^{P_1}, \overbrace{\langle\{M\}_K\rangle}^{P_2}, \overbrace{\langle cut\ R_N\ \ off\rangle}^{P_3}\}$

Int: $\{K', M'\}$

States

$\mathsf{sent}, \mathsf{received}, \mathsf{decrypted} \subseteq \mathsf{Msgs}$, initially $\mathsf{sent} = \mathsf{received} = \varnothing$ and $\mathsf{decrypted} = \{M\}$

$\mathsf{vars} \subseteq V$, $\mathsf{keys} \subseteq \mathsf{Keys}$, initially $\mathsf{vars} = V$ and $\mathsf{keys} = \{\, K, K_{SR_i} \mid i \in [N] \,\}$

Transitions

P_1
Pre: $P_1 \notin \mathsf{sent} \wedge \{\, K, K_{SR_i} \mid i \in [N] \,\} \subseteq \mathsf{keys} \wedge \mathsf{vars} = V$
Eff: $\mathsf{sent} := \mathsf{sent} \cup \{P_1\}$

P_2
Pre: $P_2 \notin \mathsf{sent} \wedge P_1 \in \mathsf{sent} \wedge M \in \mathsf{decrypted} \wedge K \in \mathsf{keys}$
Eff: $\mathsf{sent} := \mathsf{sent} \cup \{P_2\}$

P_3
Pre: $P_3 \notin \mathsf{sent} \wedge P_2 \in \mathsf{sent}$
Eff: $\mathsf{sent} := \mathsf{sent} \cup \{P_3\}$

K'
Pre: $K' \notin \mathsf{keys} \wedge P_3 \in \mathsf{sent} \wedge v_N \in \mathsf{vars}$
Eff: $\mathsf{keys} := \mathsf{keys} \cup \{K'\}$

P_4
Pre: $P_4 \notin \mathsf{received} \wedge P_3 \in \mathsf{sent}$
Eff: $\mathsf{received} := \mathsf{received} \cup \{P_4\}$

M'
Pre: $M' \notin \mathsf{decrypted} \wedge P_4 \in \mathsf{received} \wedge K' \in \mathsf{keys}$
Eff: $\mathsf{decrypted} := \mathsf{decrypted} \cup \{M'\}$

The output behaviour $\mathbf{B}_{\mathcal{T}_S}^{\Sigma_{out}}$ of $\mathcal{T}_S$ consists of all prefixes of $P_1P_2P_3$, its input behaviour $\mathbf{B}_{\mathcal{T}_S}^{\Sigma_{inp}}$ of all prefixes of P_4 and its internal behaviour $\mathbf{B}_{\mathcal{T}_S}^{\Sigma_{int}}$ of all prefixes of $K'M'$.

$\mathcal{T}_{R_1}$: group member 1

Actions

Inp: $\{\overbrace{\langle\{K, v_{\setminus 1}\}_{K_{SR_1}}, \ldots, \{K, v_{\setminus N}\}_{K_{SR_N}}\rangle}^{P_1}, \overbrace{\langle\{M\}_K\rangle}^{P_2}, \overbrace{\langle cut\ R_N\ \ off\rangle}^{P_3}\}$ Out: $\{\overbrace{\langle\{M'\}_{K'}\rangle}^{P_4}\}$

Int: $\{K, K', M, M'\}$

States

$\mathsf{sent}, \mathsf{received}, \mathsf{decrypted} \subseteq \mathsf{Msgs}$, initially $\mathsf{sent} = \mathsf{received} = \varnothing$ and $\mathsf{decrypted} = \{M'\}$

$\mathsf{vars} \subseteq V$, $\mathsf{keys} \subseteq \mathsf{Keys}$, initially $\mathsf{vars} = \varnothing$ and $\mathsf{keys} = \{K_{SR_1}\}$

Transitions

P_1
Pre: $P_1 \notin \mathsf{received}$
Eff: $\mathsf{received} := \mathsf{received} \cup \{P_1\}$

P_3
Pre: $P_3 \notin \mathsf{received} \wedge P_2 \in \mathsf{received}$
Eff: $\mathsf{received} := \mathsf{received} \cup \{P_3\}$

K
Pre: $K \notin \mathsf{keys} \wedge P_1 \in \mathsf{received} \wedge K_{SR_1} \in \mathsf{keys}$
Eff: $\mathsf{keys} := \mathsf{keys} \cup \{K\}, \mathsf{vars} := \mathsf{vars} \cup v_{\setminus 1}$

K'
Pre: $K' \notin \mathsf{keys} \wedge P_3 \in \mathsf{received} \wedge v_N \in \mathsf{vars}$
Eff: $\mathsf{keys} := \mathsf{keys} \cup \{K'\}$

P_2
Pre: $P_2 \notin \mathsf{received} \wedge P_1 \in \mathsf{received}$
Eff: $\mathsf{received} := \mathsf{received} \cup \{P_2\}$

P_4
Pre: $P_4 \notin \mathsf{sent} \wedge M' \in \mathsf{decrypted} \wedge K' \in \mathsf{keys}$
Eff: $\mathsf{sent} := \mathsf{sent} \cup \{P_4\}$

M
Pre: $M \notin \mathsf{decrypted} \wedge P_2 \in \mathsf{received} \wedge K \in \mathsf{keys}$
Eff: $\mathsf{decrypted} := \mathsf{decrypted} \cup \{M\}$

M'
Pre: $P_4 \in \mathsf{sent} \wedge M' \in \mathsf{decrypted}$
Eff: $\mathsf{decrypted} := \mathsf{decrypted} - \{M'\}$

The input behaviour $\mathbf{B}^{\Sigma_{inp}}_{\mathcal{T}_{R_1}}$ of $\mathcal{T}_{R_1}$ consists of all prefixes of $P_1P_2P_3$, its output behaviour $\mathbf{B}^{\Sigma_{out}}_{\mathcal{T}_{R_1}}$ of all prefixes of P_4 and its internal behaviour $\mathbf{B}^{\Sigma_{int}}_{\mathcal{T}_{R_1}}$ of all prefixes of $KMK'M'$.

$\mathcal{T}_{R_i}$: group member i, $2 \leq i \leq N$

Actions
Inp: $\{\overbrace{\langle\{K, v_{\setminus 1}\}_{K_{SR_1}}, \ldots, \{K, v_{\setminus N}\}_{K_{SR_N}}\rangle}^{P_1}, \overbrace{\langle\{M\}_K\rangle}^{P_2}, \overbrace{\langle cut\ R_N\ \ off\rangle}^{P_3}, \overbrace{\langle\{M'\}_{K'}\rangle}^{P_4}\}$ Out: $\varnothing$
Int: $\{K, K', M, M'\}$

States
$\mathsf{received}, \mathsf{decrypted} \subseteq \mathsf{Msgs}$, all initially $\varnothing$
$\mathsf{vars} \subseteq V$, $\mathsf{keys} \subseteq \mathsf{Keys}$, initially $\mathsf{vars} = \varnothing$ and $\mathsf{keys} = \{K_{SR_i}\}$

Transitions

P_1
Pre: $P_1 \notin \mathsf{received}$
Eff: $\mathsf{received} := \mathsf{received} \cup \{P_1\}$

P_2
Pre: $P_2 \notin \mathsf{received} \wedge P_1 \in \mathsf{received}$
Eff: $\mathsf{received} := \mathsf{received} \cup \{P_2\}$

P_3
Pre: $P_3 \notin \mathsf{received} \wedge P_2 \in \mathsf{received}$
Eff: $\mathsf{received} := \mathsf{received} \cup \{P_3\}$

P_4
Pre: $P_4 \notin \mathsf{received} \wedge P_3 \in \mathsf{received}$
Eff: $\mathsf{received} := \mathsf{received} \cup \{P_4\}$

K
Pre: $K \notin \mathsf{keys} \wedge P_1 \in \mathsf{received} \wedge K_{SR_i} \in \mathsf{keys}$
Eff: $\mathsf{keys} := \mathsf{keys} \cup \{K\}, \mathsf{vars} := \mathsf{vars} \cup v_{\setminus i}$

M
Pre: $M \notin \mathsf{decrypted} \wedge P_2 \in \mathsf{received} \wedge K \in \mathsf{keys}$
Eff: $\mathsf{decrypted} := \mathsf{decrypted} \cup \{M\}$

K'	M'
Pre: $K' \notin \mathsf{keys} \wedge P_3 \in \mathsf{received} \wedge v_N \in \mathsf{vars}$	Pre: $M' \notin \mathsf{decrypted} \wedge P_4 \in \mathsf{received} \wedge K' \in \mathsf{keys}$
Eff: $\mathsf{keys} := \mathsf{keys} \cup \{K'\}$	Eff: $\mathsf{decrypted} := \mathsf{decrypted} \cup \{M'\}$

For each $\mathcal{T}_{R_i}$, $i \in [N-1]$, its input behaviour $\mathbf{B}^{\Sigma_{inp}}_{\mathcal{T}_{R_i}}$ consists of all prefixes of $P_1P_2P_3P_4$, it has no nonempty output behaviour and its internal behaviour $\mathbf{B}^{\Sigma_{int}}_{\mathcal{T}_{R_i}}$ consists of all prefixes of $KMK'M'$. R_N, on the other hand, has the same input and output behaviour, but its internal behaviour $\mathbf{B}^{\Sigma_{int}}_{\mathcal{T}_{R_N}}$ consists of all prefixes of KM.

We enforce maximal synchronization between the component automata. Hence, the max-ai team automaton over $\{\, \mathcal{T}_S, \mathcal{T}_{R_i} \mid i \in [N] \,\}$, denoted by $\mathcal{T}_{CV}$, is defined as

$$\mathcal{T}_{CV} = |||\, \{\, \mathcal{T}_S, \mathcal{T}_{R_i} \mid i \in [N] \,\},$$

which formalizes the run of the complementary variable approach to the N-Root/Leaf pairwise keys protocol described in the previous section. Again, the max-ai synchronization of a set of component automata naturally models a broadcast communication.

5.2.2. Analyzing the Complementary Variable Approach to the N-Root/Leaf Pairwise Keys Protocol with Team Automata

In this section we use the insecure communication scenario for team automata developed in Section 3, together with a notion of the knowledge of an automaton introduced in [15], to show that leaf R_N, after it has left the group, can no longer "listen" (*i.e.,* it can receive but not decrypt the messages) to further communication within the group. Not even if, contrary to the specification of the previous section, R_N behaves as a malicious intruder rather than as a well-behaving member.

As we have done before, we assume the presence of a cryptosystem (defined by a single step derivation operator $\vdash$, with $\vdash^*$ as its transitive and reflexive closure) through which we can obtain a deduction set $\mathsf{D}(\phi) = \{\, m \mid \phi \vdash^* m\}$ of messages that can be deduced from an original set ϕ. In [15], the initial knowledge of an automaton $\mathcal{T}_A$ is bound to a specific set of messages ϕ. This informally means that the automaton should be able to produce, by means of only its internal functioning, at most the messages contained in $\mathsf{D}(\phi)$. More specifically, when considered as a stand-alone component, $\mathcal{T}_A$ can only execute output actions that belong to $\mathsf{D}(\phi)$. This initial knowledge ϕ can only be increased to a set ϕ' as the result of messages that the automaton receives during a run of the protocol under consideration. Accordingly, the automaton's knowledge then becomes at most $\mathsf{D}(\phi')$. Here we indirectly use this notion of an automaton's knowledge to prove that secrecy is preserved in the particular run of the complementary variable approach to the N-Root/Leaf pairwise keys protocol that we decribed in Section 5.2.

We start by describing the insecure scenario we use. To this aim, we instantiate the general insecure scenario for team automata sketched in Fig. 1 as follows. We consider the intruder $\mathcal{T}_X$ to be instantiated as the departing leaf R_N. This is justified because we intend to demonstrate that R_N cannot decrypt the secret message M' (that appears encrypted in P_4) once R_N has left the multicast group; not even when

R_N is behaving as a malicious intruder. We consider the initiator $\mathcal{T}_S$ to be the root S and the responder $\mathcal{T}_R$ to be a slightly modified version of R_1 (*i.e.,* the one that multicasts the message intended to be proved secret w.r.t. the departing member). Indeed, for analysis purposes, we assume R_1 to be capable of receiving also the decrypted secret message M' over the insecure connection and, whenever it does, then it immediately reveals M' to the outside as output $\texttt{Reveal}'(M')$. Moreover, R_1 does not reveal any other messages.

We have thus described the max-ai team automaton $\mathcal{T}_I$ which, to its environment, appears as a black box with the special output $\{\texttt{Reveal}'\}$. In order to analyze secrecy, we compare the observational behaviour of $\mathcal{T}_I$ with $\alpha_{sec}(\mathcal{T}_P)$, which is the expected (correct) observational behaviour of $\mathcal{T}_P$ w.r.t. secrecy. Clearly, $\alpha_{sec}(\mathcal{T}_P) = \varnothing$ because the intruder $\mathcal{T}_X$ is not present in $\mathcal{T}_P$ and without the presence of $\mathcal{T}_X$ it is impossible for M' to be transmitted in clear over the insecure connection. As a consequence, $\texttt{Reveal}'(M')$ will never never be observed in the expected (correct) observational behaviour of $\mathcal{T}_P$ w.r.t. secrecy. The proof of secrecy can now be expressed by the following theorem, where obviously $\mathsf{C} = \{\texttt{Eve}, \texttt{Eve}'\}$ as usual.

Theorem 7 $\mathcal{T}_P \in GNDC_{\subseteq}^{\alpha_{sec}(\mathcal{T}_P),\mathsf{C}}$.

Proof. To prove the statement we apply Corollary 1. In order to do so, it suffices to show that $\mathbf{O}^{\mathsf{C}}_{\mathrm{hide}_{\mathsf{C}}(|||\{\mathcal{T}_P, Top_{\mathsf{C}}^{\phi}\})} \subseteq \alpha_{sec}(\mathcal{T}_P) = \varnothing$, in which ϕ is the appropriately instantiated initial knowledge of Top_{C}^{ϕ}.

Since we consider the moment in the run of the protocol when R_N has just left the group, *i.e.,* R_1 is about to multicast P_4 to all leaves, we equip Top_{C}^{ϕ} with the initial knowledge $\phi = \{K_{SR_N}, P_1, P_2, P_3\}$. We do so because this is exactly the knowledge that every R_i, $i \in [N]$ has after R_N has left the group. Of course we furthermore assume that Top_{C}^{ϕ}, like any of the leaves, is able to generate the new key K' by applying the aforementioned appropriate function f to the pair of group key K and variable v_N (recall that $f(K, v_N) = K'$).

To prove that $\mathbf{O}^{\mathsf{C}}_{\mathrm{hide}_{\mathsf{C}}(|||\{\mathcal{T}_P, Top_{\mathsf{C}}^{\phi}\})} \subseteq \varnothing$, we evaluate how the knowledge of Top_{C}^{ϕ} evolves during protocol execution. This allows us to evaluate whether Top_{C}^{ϕ} is able to inject M' into $\mathcal{T}_P$ and, as a consequence, whether $|||\{\mathcal{T}_P, Top_{\mathsf{C}}^{\phi}\}$ may reveal M' to the outside. If it cannot, then of course neither can $\mathcal{T}_I$, which we recall to be instantiated as composed over $\mathcal{T}_P$ and R_N (rather than Top_{C}^{ϕ}).

Since $K_{SR_N} \in \phi$, it follows that Top_{C}^{ϕ} is able to decrypt both K and $v_{\setminus N} = \{v_1, \ldots, v_{N-1}\}$ from $P_1 \in \phi$, *i.e.,* $\{K, v_1, \ldots, v_{N-1}\} \subseteq \mathsf{D}(\phi)$. The only way Top_{C}^{ϕ} can further extend ϕ is by eavesdropping all the messages that are transmitted over the insecure connection.

Now assume that R_1 actually transmits P_4 to S over the insecure connection. Then the knowledge of $Top_{\mathsf{C}}^{\phi'}$ becomes $\phi' = \phi \cup \{P_4\}$. In order for $Top_{\mathsf{C}}^{\phi'}$ to generate the new key K' by using the fact that $f(K, v_N) = K'$, it must be the case that $v_N \in \mathsf{D}(\phi')$. The only way $Top_{\mathsf{C}}^{\phi'}$ can achieve this, however, is by eavesdropping variable v_N from the insecure connection. But v_N is never transmitted in clear over

the insecure connection. Therefore, $v_N \notin \mathsf{D}(\phi')$ and thus $Top_{\mathsf{C}}^{\phi'}$ cannot generate K' and, as a consequence, neither decrypt M' from P_4—let alone inject M' into $\mathcal{T}_P$. Hence $|||\{\mathcal{T}_P, Top_{\mathsf{C}}^{\phi}\}$ does not reveal M' and thus $\mathbf{O}^{\mathsf{C}}_{\mathrm{hide}_{\mathsf{C}}(|||\{\mathcal{T}_P, Top_{\mathsf{C}}^{\phi}\})} = \varnothing$. □

We have thus shown that secrecy is preserved in the particular run of the complementary variable approach to the N-Root/Leaf pairwise keys protocol that we considered: despite its communication with an intruder over an insecure connection, R_1 does not reveal M', *i.e.,* the intruder has not been able to decrypt M' from packet P_4.

Note that, as usual, our analysis only holds under the hypotheses that everyone correctly follows the protocol (*e.g.,* sets $\{v_{\setminus i}\}$ are correctly constructed), there is no collusion among the parties (a collusion occurs, *e.g.,* when someone discloses the complementary variable v_i to leaf R_i) and K'—as well as all the other secret keys involved in the protocol—is adequately chosen in order to avoid an overlap with some plaintext message. These hypotheses are assumed also in [44], thus we do not request any additional requisites w.r.t. the original protocol.

Finally, it must be noted that what we have proved secrecy in only one particular run of the complementary variables approach to the N-Root/Leaf pairwise keys protocol. It is clear that a tool is needed in which the user can specify and automatically verify secrecy in a variety of runs of this protocol. We will come back to this in the next section.

6. Conclusions and Future Work

In this article we have presented an exploration into the use of team automata for the analysis of security properties of communication protocols. More precisely, we have defined an insecure communication scenario for team automata and we have reformulated the GNDC schema in terms of team automata. Moreover, we have described two analysis strategies for the resulting setup which can be used to verify certain security properties in any communication protocol that can be modelled by the insecure communication scenario. To demonstrate this, we have performed two case studies in which we show how our framework can be used to prove integrity and secrecy in two different settings. While these case studies serve as mere toy examples, the use of the compositional analysis strategy to decompose an analysis into smaller, more manageable pieces to analyze, naturally improves the feasibility of analyzing large, complex protocols.

It remains to be seen how to extend the analysis strategies to the verification of security properties that cannot be expressed in terms of the behaviour of team automata, in particular non-safety properties. More importantly, the current pen-and-paper proofs are not enough. We need a tool in which the user can specify and automatically verify security properties in communication protocols modelled by team automata if we want our approach to become successful in practice. As said before, this is a goal for the future. Since team automata are an extension of I/O automata, the *IOA Language and Toolset* [24] may be of help when trying to achieve this goal. This latter framework provides tool support to define I/O automata and to validate their properties (through theorem proving, model checking and simulation).

References

[1] M. H. ter Beek, *Team Automata—A Formal Approach to the Modeling of Collaboration Between System Components*. PhD thesis, Leiden Institute of Advanced Computer Science, Leiden University, 2003.

[2] M. H. ter Beek, C. A. Ellis, J. Kleijn, G. Rozenberg, Team Automata for Spatial Access Control. In: *Proceedings ECSCW'01*. Kluwer, 2001, 59–77.

[3] M. H. ter Beek, C. A. Ellis, J. Kleijn, G. Rozenberg, Synchronizations in team automata for groupware systems. *Computer Supported Cooperative Work—The Journal of Collaborative Computing* **12, 1** (2003), 21–69.

[4] M. H. ter Beek, J. Kleijn, Team Automata Satisfying Compositionality. In: *Proceedings FME'03*. LNCS 2805, Springer, 2003, 381–400.

[5] M. H. ter Beek, J. Kleijn, Modularity for Teams of I/O Automata. *Information Processing Letters* **95, 5** (2005), 487–495.

[6] M. H. ter Beek, G. Lenzini, M. Petrocchi, Team Automata for Security Analysis of Multicast/Broadcast Communication. Techn. Rep. 2003-TR-13, Istituto di Scienza e Tecnologie dell'Informazione, CNR, 2003. Presented at WISP'03: *Workshop on Issues in Security and Petri nets*, no formal proceedings.

[7] M. H. ter Beek, G. Lenzini, M. Petrocchi, Team Automata for Security Analysis. Techn. Rep. TR-CTIT-04-13, Centre for Telematics and Information Technology, Twente University, 2004. Presented at DIMACS'04 Workshop: *Security Analysis of Protocols*, no formal proceedings.

[8] M. H. ter Beek, G. Lenzini, M. Petrocchi, Team Automata for Security – A Survey –. In: *Proceedings SecCo'04*. ENTCS 128, 5, Elsevier, 2005, 105–119.

[9] B. Bell, L. La Padula, Secure Computer Systems—Unified Exposition and Multics Interpretation. Techn. Rep. ESD-TR-75-306, MITRE MTR-2997, 1976.

[10] C. Cachin, J. Camenisch, J. Kilian, J. Müller, One-Round Secure Computation and Secure Autonomous Mobile Agents. In: *Proceedings ICALP'00*. LNCS 1853, Springer, 2000, 512–523.

[11] R. Canetti, J. A. Garay, G. Itkis, D. Micciancio, M. Naor, B. Pinkas, Multicast Security: A Taxonomy and Some Efficient Constructions. In: *Proceedings INFOCOM'99*. IEEE Press, 1999, 708–716.

[12] E. M. Clarke, S. Jha, W. Marrero, Verifying Security Protocols with Brutus. *ACM Transactions on Software Engineering and Methodology* **9, 4** (2000), 443–487.

[13] W. Diffie, M. Hellman, New Directions in Cryptography. *IEEE Transactions on Information Theory* **22, 6** (1976), 644–656.

[14] D. Dolev, A. Yao, On the security of public-key protocols. *IEEE Transactions on Information Theory* **2, 29** (1983), 198–208.

[15] L. Egidi, M. Petrocchi, Modelling a Secure Agent with Team Automata. In: *Proceedings VODCA'04*. ENTCS 142, Elsevier, 2006, 111–127.

[16] C. A. Ellis, Team Automata for Groupware Systems. In: *Proceedings GROUP'97*. ACM Press, 1997, 415–424.

[17] R. Focardi, R. Gorrieri, A Classification of Security Properties for Process Algebras. *Journal of Computer Security* **3, 1** (1995), 5–34.

[18] R. Focardi, R. Gorrieri, The Compositional Security Checker: A Tool for the Verification of Information Flow Security Properties. *IEEE Transactions on Software Engineering* **23, 9** (1997), 550–571.

[19] R. Focardi, R. Gorrieri, Classification of Security Properties—Part I: Information Flow. In: *FOSAD'01 Tutorial Lectures*. LNCS 2171, Springer, 2001, 331–396.

[20] R. Focardi, R. Gorrieri, F. Martinelli, Non Interference for the Analysis of Cryptographic Protocols. In: *Proceedings ICALP'00*. LNCS 1853, Springer, 2000, 354–372.

[21] R. Focardi, R. Gorrieri, F. Martinelli, Secrecy in Security Protocols as Non Interference. In: *Proceedings Workshop on secure architectures and information flow*. ENTCS 32, Elsevier, 2000.

[22] R. Focardi, R. Gorrieri, F. Martinelli, Classification of Security Properties—Part II: Network Security. In: *FOSAD 2001-02 Tutorial Lectures*. LNCS 2946, Springer, 2004, 139–185.

[23] R. Focardi, F. Martinelli, A Uniform Approach for the Definition of Security Properties. In: *Proceedings FM'99*. LNCS 1708, Springer, 1999, 794–813.

[24] S. J. Garland, N. A. Lynch, The IOA Language and Toolset—Support for Designing, Analyzing, and Building Distributed Systems. Techn. Rep. MIT/LCS/TR-762, MIT, 1998.

[25] J. A. Goguen, J. Meseguer, Security Policy and Security Models. In: *Proceedings S&P'82*. IEEE Press, 1982, 11–20.

[26] R. Gorrieri, F. Martinelli, A simple framework for real-time cryptographic protocol analysis with compositional proof rules. *Science of Computer Programming* **50, 1–3** (2004), 23–49.

[27] R. Gorrieri, F. Martinelli, M. Petrocchi, A. Vaccarelli, Compositional Verification of Integrity for Digital Stream Signature Protocols. In: *Proceedings ACSD'03*. IEEE Press, 2003, 142–149.

[28] S. Gürgens, P. Ochsenschläger, C. Rudolph, Role Based Specification and Security Analysis of Cryptographic Protocols Using Asynchronous Product Automata. In: *Proceedings DEXA'02*. IEEE Press, 2002, 473–482.

[29] J. Kleijn, Team Automata for CSCW – A Survey –. In: *Petri Net Technology for Communication-Based Systems—Advances in Petri Nets*. LNCS 2472, Springer, 2003, 295–320.

[30] R. Lanotte, A. Maggiolo-Schettini, A. Troina, Weak Bisimulation for Probabilistic Timed Automata and Applications to Security. In: *Proceedings SEFM'03*. IEEE Press, 2003, 34–43.

[31] R. Lanotte, A. Maggiolo-Schettini, A. Troina, Decidability Results for Parametric Probabilistic Transition Systems with an Application to Security. In: *Proceedings SEFM'04*. IEEE Press, 2004, 114–121.

[32] R. Lanotte, A. Maggiolo-Schettini, A. Troina, Information Flow Analysis for Probabilistic Timed Automata. In: *Proceedings FAST'04*. IFIP 173, Springer, 2005, 13–26.

[33] G. Lenzini, *Integration of Analysis Techniques in Security and Fault-Tolerance*. PhD thesis, Centre for Telematics and Information Technology, University of Twente, 2005.

[34] G. Lenzini, S. Gnesi, D. Latella, Spider: a Security Model Checker. In: *Proceedings FAST'03*, 2003, 163–180.

[35] G. Lowe, Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR. In: *Proceedings TACAS'96*, LNCS 1055, Springer, 1996, 147–166.

[36] N. A. Lynch, I/O Automaton Models and Proofs for Shared-Key Communication Systems. In: *Proceedings CSFW'99*. IEEE Press, 1999, 14–31.

[37] N. A. Lynch, M. R. Tuttle, An Introduction to Input/Output Automata. *CWI Quarterly* **2, 3** (1989), 219–246. Also published as Techn. Memo MIT/LCS/TM-373, MIT, 1988.

[38] F. Martinelli, M. Petrocchi, A. Vaccarelli, Compositional Verification of Secure Streamed Data: A Case Study with EMSS. In: *Proceedings ICTCS'03*. LNCS 2841, Springer, 2003, 383–396.

[39] R. M. Needham, M. D. Schroeder, Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM* **21, 12** (1978), 993–999.

[40] D. von Oheimb, Interacting State Machines—A Stateful Approach to Proving Security. In: *Proceedings FASec'02*. LNCS 2629, Springer, 2003, 15–32.

[41] D. von Oheimb, V. Lotz, Formal Security Analysis with Interacting State Machines. In: *Proceedings ESORICS'02*. LNCS 2502, Springer, 2002, 212–228.

[42] A. Perrig, R. Canetti, J. D. Tygar, D. X. Song, Efficient Authentication and Signing of Multicast Streams over Lossy Channels. In: *Proceedings S&P'00*. IEEE Press, 2000, 56–73.

[43] M. Petrocchi, *Aspects of Modeling and Verifying Secure Procedures*. PhD thesis, Dipartimento di Ingegneria dell'Informazione, Pisa University, 2005.

[44] D. Wallner, E. Harder, R. Agee, Key Management for Multicast: Issues and Architectures. IETF, RFC 2627, June 1999.
URL: `http://www.ietf.org/rfc/rfc2627.txt`