Team Automata for Security Analysis

Maurice H. ter Beek
Istituto di Scienza e Tecnologie dell'Informazione, CNR
Area della Ricerca di Pisa, Via G. Moruzzi 1, 56124 Pisa, Italy
maurice.terbeek@isti.cnr.it

Gabriele Lenzini
Department of Computer Science, University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands
lenzinig@cs.utwente.nl

Marinella Petrocchi
Istituto di Informatica e Telematica, CNR
Area della Ricerca di Pisa, Via G. Moruzzi 1, 56124 Pisa, Italy
marinella.petrocchi@iit.cnr.it

Abstract

We show that team automata (TA) are well suited for security analysis by reformulating the Generalized Non-Deducibility on Compositions (GNDC) schema in terms of TA. We then use this to show that integrity is guaranteed for a case study in which TA model an instance of the Efficient Multi-chained Stream Signature (EMSS) protocol.

1. Introduction

There is an increasing interest in using automata-based models for the formal specification and analysis of security properties of communication protocols [11, 16, 18, 21]. We continue this line of research by showing how to use team automata (TA) [3] for such security analysis.

TA form a flexible framework for modelling communication between system components [2, 3, 15]. They model the logical architecture of a system by describing it in terms of (component) automata, the role of actions, and synchronizations between these actions. The crux of composing a TA is to define the way in which its constituting component automata (CA) communicate by synchronizations. Hence there exists no unique TA given a set of CA, but rather a whole range of TA can be constructed from such a set. Which one to choose can thus be based on the specific communication protocol to be modelled. It is this freedom that sets TA apart from most other models (cf. Sect. 2.1 for a brief comparison of TA with some models of concurrency). In [5], *e.g.,* we showed how this allows TA to model multicast and broadcast communications in a very natural way.

In [5] we initiated a very preliminary investigation on defining the Generalized Non-Deducibility on Compositions (GNDC) schema in terms of TA. This is a general schema for defining and analyzing security properties, which was originally formalized in [9] for a cryptographic version of the process algebra CCS. A formal model P of a system (*e.g.,* a protocol) is said to satisfy $GNDC_{\triangleleft}^{\alpha(P)}$ if and only if P, despite the fact that it interacts with a hostile environment, appears indistinguishable (w.r.t. a notion $\triangleleft$ of external observation) from the expected (*i.e.,* correct) behaviour $\alpha(P)$ of P. By varying the parameters $\triangleleft$ and α, many security properties can be formulated and analyzed within this theory [8, 9, 10, 20].

In this article we complete our preliminary investigation by reformulating the GNDC schema in terms of TA and thus equip TA with a theory for security analysis. As an application, we verify the property of integrity for a case study in which TA model an instance of the Efficient Multi-chained Stream Signature (EMSS) protocol [22].

The article is organized as follows. In Sect. 2 we define TA. In Sect. 3 we describe a case study modelling an instance of the EMSS protocol by TA. We then describe an insecure communication scenario in Sect. 4, after which we reformulate the GNDC schema in terms of TA in Sect. 5. We then use this in Sect. 6 to analyze integrity in our case study and conclude the paper.

2. Team Automata

We begin this section by fixing some notations and terminology used throughout this article, after which we recall some definitions and results concerning TA from [3, 4].

For convenience we denote the set $\{1, \ldots, n\}$ by $[n]$. 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 $j \in [n]$, $\text{proj}_j : \prod_{i\in[n]} V_i \to V_j$ is defined by $\text{proj}_j((a_1, \ldots, a_n)) = a_j$. The powerset of a set V is denoted by 2^V.

Let Σ and Γ be two sets of symbols. Then the projection $\text{pres}_{\Sigma,\Gamma} : \Sigma \to \Gamma^*$, defined by $\text{pres}_{\Sigma,\Gamma}(a) = a$ if $a \in \Gamma$ and $\text{pres}_{\Sigma,\Gamma}(a) = \lambda$ otherwise, preserves the symbols from Γ and erases all other symbols. Whenever Σ is clear from the context, we simply write pres_Γ rather than $\text{pres}_{\Sigma,\Gamma}$.

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 *An* automaton *is a construct* $\mathcal{A} = (Q, \Sigma, \delta, I)$*, with set* Q *of* states*, set* Σ *of* actions*,* $Q \cap \Sigma = \varnothing$*, set* $\delta \subseteq Q \times \Sigma \times Q$ *of* transitions*, and set* $I \subseteq Q$ *of* initial states*.*

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

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

The Σ-behaviour of $\mathcal{A}$ is also called the behaviour of $\mathcal{A}$, in which case Σ may be discarded from the notation. Let $a \in \Sigma$. The set δ_a of *a-transitions* of $\mathcal{A}$ is defined as $\delta_a = \{(q, q') \mid (q, a, q') \in \delta\}$. Finally, note that behavioural inclusion defines a preorder relation on automata.

A TA is composed of component automata (CA), which are ordinary automata without final states and with a distinction of their sets of actions into input, output and internal actions. Their internal actions have strictly local visibility and cannot be used for communication with other CA. Their input and output actions are observable by other CA and are used for communication between CA.

Definition 2 *A* component automaton (CA) *is a construct* $\mathcal{C} = (Q, (\Sigma^\mathcal{C}_{inp}, \Sigma^\mathcal{C}_{out}, \Sigma^\mathcal{C}_{int}), \delta, I)$*, with* underlying automaton $(Q, \Sigma^\mathcal{C}_{inp} \cup \Sigma^\mathcal{C}_{out} \cup \Sigma^\mathcal{C}_{int}, \delta, I)$ *and pairwise disjoint sets* $\Sigma^\mathcal{C}_{inp}$ *of* input*,* $\Sigma^\mathcal{C}_{out}$ *of* output*, and* $\Sigma^\mathcal{C}_{int}$ *of* internal actions*.*

$\Sigma^\mathcal{C}$ denotes the set $\Sigma^\mathcal{C}_{inp} \cup \Sigma^\mathcal{C}_{out} \cup \Sigma^\mathcal{C}_{int}$ of actions of $\mathcal{C}$ and $\Sigma^\mathcal{C}_{ext}$ denotes its set $\Sigma^\mathcal{C}_{inp} \cup \Sigma^\mathcal{C}_{out}$ of external actions. We discard $\mathcal{C}$ from these notations when no confusion can arise.

For the sequel we let $\mathcal{S} = \{\mathcal{C}_i \mid i \in [n]\}$ be an arbitrary but fixed set of CA 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 and set $\Sigma_{i,ext} = \Sigma_{i,inp} \cup \Sigma_{i,out}$ of external actions.

When composing TA over $\mathcal{S}$, the internal actions of the CA in $\mathcal{S}$ must be private, *i.e.*, $\forall\, i \in [n] : \Sigma_{i,int} \cap \bigcup_{j\in([n]-\{i\})} \Sigma_j = \varnothing$. Such an $\mathcal{S}$ is called a *composable system*. For the sequel we let $\mathcal{S}$ be a composable system.

The state space of a TA composed over $\mathcal{S}$ is the product of the state spaces of the CA from $\mathcal{S}$. Its actions, consequently, are uniquely determined by the actions of the CA from $\mathcal{S}$. Each action that is output (internal) for one or more of the CA becomes an output (internal) action of the TA. Hence an action that is an output action of one CA and also an input action of another CA, is considered an output action of the TA. The input actions of the CA that do not occur at all as an output action of any of the CA, become the input actions of the TA. The transitions of the TA, finally, are based on but not fixed by those of the CA from $\mathcal{S}$ by allowing certain synchronizations, while excluding others.

Definition 3 *Let* $a \in \bigcup_{i\in[n]} \Sigma_i$*. The set* $\Delta_a(\mathcal{S})$ *of* synchronizations *of* a *is defined as* $\Delta_a(\mathcal{S}) = \{(q, q') \in \prod_{i\in[n]} Q_i \times \prod_{i\in[n]} Q_i \mid [\,\exists\, j \in [n] : \mathit{proj}_j{}^{[2]}(q, q') \in \delta_{j,a}\,] \wedge [\,\forall\, i \in [n] : [\,\mathit{proj}_i{}^{[2]}(q, q') \in \delta_{i,a}\,] \vee [\,\mathit{proj}_i(q) = \mathit{proj}_i(q')\,]\,]\}$*.*

$\Delta_a(\mathcal{S})$ thus consists of all possible combinations of a-transitions of CA from $\mathcal{S}$, with all non-participating CA remaining idle. It is explicitly required that in every synchronization at least one CA participates. The state change of a TA over $\mathcal{S}$ is defined by the local state changes of the CA from $\mathcal{S}$ participating in the action of the TA being executed. Hence, when defining a TA, a specific subset of $\Delta_a(\mathcal{S})$ must be chosen for each action a. This enforces a certain kind of communication between the CA constituting the TA.

Definition 4 *A* team automaton (TA) *over* $\mathcal{S}$ *is a construct* $\mathcal{T} = (Q, (\Sigma^\mathcal{T}_{inp}, \Sigma^\mathcal{T}_{out}, \Sigma^\mathcal{T}_{int}), \delta, I)$*, with* $Q = \prod_{i\in[n]} Q_i$*,* $\Sigma^\mathcal{T}_{inp} = (\bigcup_{i\in[n]} \Sigma_{i,inp}) - \Sigma^\mathcal{T}_{out}$*,* $\Sigma^\mathcal{T}_{out} = \bigcup_{i\in[n]} \Sigma_{i,out}$*,* $\Sigma^\mathcal{T}_{int} = \bigcup_{i\in[n]} \Sigma_{i,int}$*,* $\delta \subseteq Q \times \Sigma^\mathcal{T} \times Q$*, where* $\Sigma^\mathcal{T} = \Sigma^\mathcal{T}_{inp} \cup \Sigma^\mathcal{T}_{out} \cup \Sigma^\mathcal{T}_{int}$*, is such that* $\{(q, q') \mid (q, a, q') \in \delta\} \subseteq \Delta_a(\mathcal{S})$*, for all* $a \in \Sigma^\mathcal{T}$*, and* $\{(q, q') \mid (q, a, q') \in \delta\} = \Delta_a(\mathcal{S})$*, for all* $a \in \Sigma^\mathcal{T}_{int}$*, and* $I = \prod_{i\in[n]} I_i$*.*

Each choice of synchronizations thus defines a TA. It is important to observe that *every TA is again a CA*, which in its turn can be used as a CA in an *iteratively* composed TA. In this way one can construct, *e.g.*, a TA $\mathcal{T}'$ over the composable system $\{\mathcal{T}'', \mathcal{C}_3\}$, where $\mathcal{T}''$ is a TA composed over the composable system $\{\mathcal{C}_1, \mathcal{C}_2\}$. TA can thus be used as building blocks. In order to do so, two important issues must be dealt with.

First it may be necessary to internalize certain external actions of a TA before using this TA as a building block, in order to prohibit the use of these actions on a higher level of the construction (we moreover assume that such actions are indexed in order to satisfy composability).

Definition 5 *Let $\mathcal{T} = (Q, (\Sigma_{inp}, \Sigma_{out}, \Sigma_{int}), \delta, I)$ be a TA 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 hide$_\Gamma(\mathcal{T})$, the subset Γ of external actions of $\mathcal{T}$ have thus become unobservable for other TA by turning them into internal actions. Without formally defining renaming, we assume that these actions are indexed by the TA in order to guarantee composability.

Secondly, it must be possible to construct unique TA of a specified type. In [3] several fixed strategies for choosing the synchronizations of a TA were defined, each leading to a uniquely defined TA. These strategies fix the synchronizations of a TA 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 TA. Such subsets are referred to as *predicates* for a. Once predicates have been chosen for all actions, the TA over $\mathcal{S}$ defined by such a predicate is unique.

Definition 6 *Let $a \in \bigcup_{i\in[n]} \Sigma_i$. The predicate* is-ai *for a in $\mathcal{S}$, denoted by $\mathcal{R}_a^{ai}(\mathcal{S})$, 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 predicate $\mathcal{R}_a^{ai}(\mathcal{S})$ thus contains *all and only* those a-transitions from $\Delta_a(\mathcal{S})$ in which every CA with a as an action participates. Hence the TA over $\mathcal{S}$ defined by this predicate is the unique TA in which any execution of a sees the participation of all CA having a in their set of actions.

Definition 7 *Let $\mathcal{R}^{ai} = \{\mathcal{R}_a^{ai}(\mathcal{S}) \mid a \in \bigcup_{i\in[n]} \Sigma_i\}$. Then $\mathcal{T} = (Q, (\Sigma_{inp}, \Sigma_{out}, \Sigma_{int}), \delta, I)$ is* the max-ai TA *over $\mathcal{S}$, denoted by $|||\,\mathcal{S}$, if $\delta_a = \mathcal{R}_a^{ai}(\mathcal{S})$, for all $a \in \bigcup_{i\in[n]} \Sigma_i$.*

Remark 1 ([3]) *The behaviour of an iteratively composed max-ai TA is equal to that of the max-ai TA over the underlying CA,* i.e., *continuing our above example: if $\mathcal{T}'$ and $\mathcal{T}''$ are the max-ai TA over $\{\mathcal{T}'', \mathcal{C}_3\}$ and $\{\mathcal{C}_1, \mathcal{C}_2\}$, resp., and $\mathcal{T}$ is the max-ai TA over $\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$, then $\mathbf{B}_{\mathcal{T}'} = \mathbf{B}_{\mathcal{T}}$.*

Remark 2 *If $\{\mathcal{T}, \mathcal{T}\}$ is a composable system, then trivially $\mathbf{B}_{|||\,\{\mathcal{T},\mathcal{T}\}} = \mathbf{B}_{\mathcal{T}}$.*

A TA is said to satisfy compositionality if its behaviour can be described in terms of that of its constituting CA, *i.e.,* when the traces forming the behaviour of a set of CA can be *shuffled* in such a way that the traces forming the behaviour of a particular TA over these CA result.

Definition 8 *The* full synchronized shuffle $\underline{||}_{\{\Delta_i \mid i\in[n]\}} L_i$ *of $L_i \subseteq \Delta_i^*$, with $i \in [n]$, is defined as $\underline{||}_{\{\Delta_i \mid i\in[n]\}} L_i = \{w \in (\bigcup_{i\in[n]} \Delta_i)^* \mid \forall\, i \in [n] : pres_{\Delta_i}(w) \in L_i\}$.*

In [4] it was shown that the construction of TA according to certain natural types of synchronization, among which the one leading to max-ai TA, guarantees compositionality.

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

2.1. TA in Relation to Other Models

TA are not an isolated model, but instead possess several features that bear a close resemblance to other models from the literature. In this section we briefly discuss the relationship between TA and two models of concurrency. To begin with, TA are an extension of *(safe) Input/Output Automata* (IOA) [19]. However, contrary to TA, IOA are input enabled (*i.e.,* in each state it must be possible to execute every input action). Furthermore there is no unique TA composed over a set of CA, but a whole range of TA—distinguishable only by their synchronizations—can be composed over this set of CA. Given a set of IOA, on the other hand, an IOA over this set is constructed according to the is-ai predicate defined above, which results in IOA that are uniquely defined by their constituents. Finally, IOA do not allow synchronizations of output actions, whereas TA do.

TA are furthermore related to Petri nets. Since the synchronizations in TA describe state changes caused by global team actions, the operational semantics in terms of TA computations is of a sequential nature and does not reflect the fact that TA are distributed systems. If we switch from global (team) actions to vectors of (component) actions, however, then we obtain the local information from which we can immediately extract exactly which of the CA participate in a team synchronization. The resulting *vector TA* visualize the potential concurrency within TA and thus relate TA to Petri nets. In fact, in [15] it was shown that a subset of vector TA can be translated to a particular model of vector-labelled Petri nets from the framework of Vector Controlled Concurrent Systems (VCCSs) [13], viz. to the Individual Token Net Controllers (ITNCs) [14].

The idea underlying VCCSs is to use vectors both to specify the elementary synchronizations within a system and to describe its behaviour. This approach was inspired by the vector firing sequence semantics of COSY [12] and is related to the coordination of cooperating automata by synchronization on multisets [1]. Several differences remain between vector TA and ITNCs. Contrary to vector TA, ITNCs allow fundamentally different actions to synchronize. In this respect, ITNCs thus allow the modelling of more types of synchronization than (vector) TA do. However, ITNCs do not distinguish between input, output, and internal actions, which is a crucial modelling feature of (vector) TA.

3. A Case Study: the EMSS Protocol

This section shows how an instance of the EMSS protocol can be modelled in terms of TA. The EMSS protocol was introduced in [22] and is used to sign digital streams. It exploits a combination of hash functions and digital signatures and 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, the receivers are not able to communicate among each other). The protocol then requires S to send triples built from payloads (called packets) to the receivers. After an initial phase, each packet P_i contains a meaningful payload m_i (we assume that the private sender key $sk(S)$ cannot be deduced from $\{m_i \mid 0 \leq i \leq \mathit{last}\}$), 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. In this way, some robustness against packet loss is achieved. The protocol can formally be described as follows, with $2 \leq i \leq \mathit{last}$.

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

3.1. The EMSS Protocol Modelled by TA

We now show how TA can be used to model the EMSS protocol. We model the sender S by a CA $\mathcal{T}_S$ 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 CA $\mathcal{T}_R$ (we assume the internal actions of each $\mathcal{T}_R^{(i)}$ to be indexed in order to satisfy composability). $\mathcal{T}_S$ uses its private key $sk(\mathcal{T}_S)$ and a public key $pk(\mathcal{T}_S)$ to perform digital signature operations. Let Payloads denote the set $\{m_0, m_1, \ldots, m_{\mathit{last}}\}$ of meaningful payloads and let Payloads$'$ denote the set $\{m'_0, m'_1, \ldots, m'_{\mathit{last}}\}$ of variables m'_i that should contain the meaningful payloads m_i, with $0 \leq i \leq \mathit{last}$. Then $\mathcal{T}_S$ uses the hash function h : Payloads $\rightarrow$ Hashed, while $\mathcal{T}_R$ uses the hash function $\bar{h} = h$. Moreover, $\mathcal{T}_S$ uses the function $s : 2^{\mathsf{Hashed}} \rightarrow$ Signed, defined by $s(H) = H_{sk(\mathcal{T}_S)}$, to sign sets of hashed payloads with its private key $sk(\mathcal{T}_S)$, whereas $\mathcal{T}_R$ uses the function $\bar{s}$: Signed $\rightarrow \{\mathsf{true}, \mathsf{false}\}$ and the public key $pk(\mathcal{T}_S)$ to verify whether a set of hashed payloads was signed by $\mathcal{T}_S$.

We specify TA in the way IOA are commonly defined [18, 19]. The states of a TA are thus 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 a TA if the precondition of a is satisfied by q, while q' is the transformation of q defined by the effect of a. We omit the precondition (effect) of an action when it is true.

$\mathcal{T}_S$

Actions

Inp: $\varnothing$

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

Int: $\{\mathit{Hash}_i \mid 0 \leq i \leq \mathit{last}\} \cup \{\mathit{Sign}\}$

States

sent $\subseteq$ Payloads, hashed $\subseteq$ Hashed, signed $\subseteq$ Signed, all initially $\varnothing$

Transitions

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

$\mathit{Hash}_i,\ 0 \leq i \leq \mathit{last}$
Pre: $P_i \in$ sent $\wedge\ h(P_i) \notin$ hashed
Eff: hashed := hashed $\cup \{h(P_i)\}$

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

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

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

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

Clearly $\mathcal{T}_S$ has no input behaviour, while its output behaviour $\mathbf{B}_{\mathcal{T}_S}^{\Sigma_{out}}$ consists of all prefixes of $P_0 P_1 \cdots P_{\mathit{last}} P_{\mathit{sign}}$. To actually send the packets $P_0, P_1, \ldots, P_{\mathit{last}}, P_{\mathit{sign}}$ in this order, $\mathcal{T}_S$ must perform some internal computations, which is reflected by the fact that its internal behaviour $\mathbf{B}_{\mathcal{T}_S}^{\Sigma_{int}}$ consists of all prefixes of $\mathit{Hash}_0 \mathit{Hash}_1 \cdots \mathit{Hash}_{\mathit{last}} \mathit{Sign}$.

We continue with the specification of $\mathcal{T}_R$. It is capable of receiving as input behaviour all packets $P_0, P_1, \ldots, P_{\mathit{last}}, P_{\mathit{sign}}$, built over the set 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 that $\mathcal{T}_R$ 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 that $\mathcal{T}_R$ did not receive P_{i-1}. Then it verifies whether it has received P_{i-2}. If it did not, then

$\mathcal{T}_R$ concludes that it is unable to check the hashes of either P_{i-1} or P_{i-2}, so it goes on to verify whether it has received 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 these 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 sometimes check if P_{sign} has already been received to avoid a transition to take place *before* P_{sign} has actually been 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 consequently 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 $P_0 \notin$ sent prohibits this action to be executed again. For the sake of readability, we omit the addition of such preconditions to the specification of $\mathcal{T}_R$, but implicitly assume that its actions are executed only once during a computation.

$\mathcal{T}_R$

Actions

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

Out: Payloads′

Int: $\{\mathit{XtractH}_i, \mathit{XtractM}_i, \mathit{Hash}_i \mid 0 \leq i \leq \mathit{last}\} \cup \{\mathit{Verify}, \mathit{Stream}\}$

States

received,xtractedM $\subseteq$ Payloads′, xtractedH,hashed $\subseteq$ Hashed, all initially $\varnothing$, verified,send $\subseteq \{$true, false$\}$, both initially false

Transitions

$P_i,\ 0 \leq i \leq \mathit{last}$
Eff: received := received $\cup \{P_i\}$

$\mathit{XtractH}_{i,1},\ 1 \leq i \leq \mathit{last}$
Pre: $\{P_{i-1}, P_i\} \subseteq$ received
Eff: xtractedH := xtractedH $\cup \{h(P_{i-1})\}$

$\mathit{XtractH}_{i,2},\ 2 \leq i \leq \mathit{last}$
Pre: $[\,\{P_{i-2}, P_i\} \subseteq$ received $\,] \wedge [\,P_{i-1} \notin$ received $\,]$
Eff: xtractedH := xtractedH $\cup \{h(P_{i-2})\}$

P_{sign}
Eff: received := received $\cup \{P_{sign}\}$

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

$\mathit{XtractH}_{sign,1}$
Pre: $[\,\{P_{last}, P_{sign}\} \subseteq$ received $\,] \wedge [\,$verified = true$\,]$
Eff: xtractedH := xtractedH $\cup \{h(P_{last})\}$

$\mathit{XtractH}_{sign,2}$
Pre: $[\,\{P_{last-1}, P_{sign}\} \subseteq$ received $\,] \wedge [\,P_{last} \notin$ received $\,] \wedge [\,$verified = true$\,]$
Eff: xtractedH := xtractedH $\cup \{h(P_{last-1})\}$

Stream
Pre: $[\,[\,m'_{last} \in$ xtractedM $\,] \vee [\,[\,m'_{last-1} \in$ xtractedM $\,] \wedge [\,P_{last} \notin$ received $\,]\,] \wedge [\,$verified = true$\,]$
Eff: send := true

$\mathit{XtractM}_i,\ 0 \leq i \leq \mathit{last}$
Pre: $[\,h(P_i) \in$ xtractedH $\,] \wedge [\,\bar{h}(P_i) \in$ hashed $\,] \wedge [\,\bar{h}(P_i) = h(P_i)\,]$
Eff: xtractedM := xtractedM $\cup \{m'_i\}$

$\mathit{Hash}_i,\ 0 \leq i \leq \mathit{last}$
Pre: $h(P_i) \in$ xtractedH
Eff: hashed := hashed $\cup \{\bar{h}(P_i)\}$

m'_0
Pre: $[\,$send = true$\,] \wedge [\,m'_0 \in$ xtractedM $\,]$
Eff: xtractedM := xtractedM $- \{m'_0\}$

$m'_i,\ 1 \leq i \leq \mathit{last}$
Pre: $[\,$send = true$\,] \wedge [\,m'_i \in$ xtractedM $\,] \wedge [\,\{m'_k \mid 0 \leq k < i\} \cap$ xtractedM $= \varnothing\,]$
Eff: xtractedM := xtractedM $- \{m'_i\}$

Clearly the input behaviour $\mathbf{B}_{\mathcal{T}_R}^{\Sigma_{inp}}$ of $\mathcal{T}_R$ consists of all prefixes of all possible permutations of $P_0 P_1 \cdots P_{last} P_{sign}$. When $\mathcal{T}_R$ actually receives the packets $P_0, P_1, \ldots, P_{last}, P_{sign}$ in this particular order, then $\mathcal{T}_R$ 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 $\mathit{XtractH}_{1,1}\,\mathit{Hash}_0\,\mathit{XtractM}_0\,\mathit{XtractH}_{2,1}\,\mathit{Hash}_1\,\mathit{XtractM}_1 \cdots \mathit{XtractH}_{last,1}\ \mathit{Hash}_{last-1}\ \mathit{XtractM}_{last-1}\ \mathit{Verify}\ \mathit{XtractH}_{sign,1}\ \mathit{Hash}_{last}\mathit{XtractM}_{last}\mathit{Stream}$ as well other traces representing other orders of performing these internal computations. Finally, the output behaviour $\mathbf{B}_{\mathcal{T}_R}^{\Sigma_{out}}$ of $\mathcal{T}_R$ consists of all prefixes of $m'_0 m'_1 \cdots m'_{last}$.

Now the max-ai TA 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) actions of $\mathcal{T}_S$ and the $\mathcal{T}_R$'s as its output (internal) actions. The fact that its output behaviour consists of all prefixes of $P_0 P_1 \cdots P_{last} P_{sign} m'_0 m'_1 \cdots m'_{last}$ implies that it models broadcast communication. By composing other TA, we can model multicast communication and packet loss. In [5] we presented a case study dealing with such issues.

4. The Insecure Communication Scenario

In this section we describe how to adapt a generic protocol specification in order to obtain an insecure scenario in which to analyze security properties. When describing our insecure scenario we assume all actions of TA to be built over a first order signature σ, where predicate symbols are seen as communication channels and atomic formulae as messages. We assume that the function symbols in σ contain at least the ones that we have implicitly used in the previous section, *i.e.*: the symbols denoting the encryption and pairing functions, *e.g.*, $\{\_\}\_$ and $\langle \_, \_ \rangle$; the symbols denoting the hash functions, *e.g.*, $h(\_)$; those indicating the secret and public key, *e.g.*, $sk(\_)$ and $pk(\_)$. We let m, m' range over the set Messages of atomic formulae, and c, c' over the set Channels of predicate symbols. In the sequel, `Eve`, `Eve`$'$, `Pub`, `Pub`$'$, and `Reveal` will be used as particular predicate names. Hence every action will be written as $c(m)$, *i.e.*, denoting the message m sent over the channel c. Given a set $M \subseteq$ Messages of messages, we define $c(M) = \{c(m) \mid m \in M\}$. Given a set C of predicate names we define $C(M) = \{c(m) \mid m \in M,\ c \in C\}$. Finally, with a little abuse of notation, in the sequel we will write C also as a shortcut for the set $C(\mathsf{Messages})$.

We abstract from the cryptographic details concerning the operations according to which these messages can be encrypted, decrypted, hashed, etc., but we assume the presence of a cryptosystem (defined by a derivation operator $\vdash$) that implements these operations. By applying cryptographic operations from this cryptosystem to a set M of messages, a new set $\mathcal{D}(M) = \{m \mid M \vdash m\}$ of messages (usually called the *deduction set*) can be obtained. This is a standard approach in the formal analysis of cryptographic protocols of communication [6, 9, 17, 18].

In the sequel we assume a protocol specification involving only two roles, viz. an *initiator* $\mathcal{T}_S$ and a *responder* $\mathcal{T}_R$. Contrary to a direct communication between $\mathcal{T}_S$ and $\mathcal{T}_R$ as considered so far, we assume all the communication to flow through an *insecure channel* (cf. Fig. 1). This insecure channel may release some messages to an *intruder* which, in its turn, can either listen to or modify (fake) the messages passing through this channel. When verifying security properties for communication protocols, it is indeed quite common to include an additional intruder component that is supposed to be malicious and whose aim is to subvert the protocol's correct behaviour. A protocol specification is consequently considered secure w.r.t. a security property if it satisfies this property despite the presence of the intruder. Based on the approach of [18], the insecure channel and the intruder are modelled by TA $\mathcal{T}_{IC}$ and $\mathcal{T}_X$. We thus propose a framework of four types of TA:

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 channel, and
4. $\mathcal{T}_X$ plays the role of the intruder.

We do not explicitly model the TA of our framework, but we informally describe them by their interactions. More precisely, we let the initiator and the responder communicate with the insecure channel through disjoint sets of actions Σ^S_{com} and Σ^R_{com}, resp., such that a direct communication between them is impossible. The $\mathcal{T}_{IC}$, in its turn, can interact with the intruder only through a distinct set Σ^I_{com} of actions. Finally, some particular actions may be used by an honest role in order to reveal some information to the outside concerning, *e.g.*, a state reached during a run of the protocol. In Fig. 1 an example of such a scenario is given for protocols with a unique sender and a unique receiver. This scenario thus suits a variety of stream signature protocols.

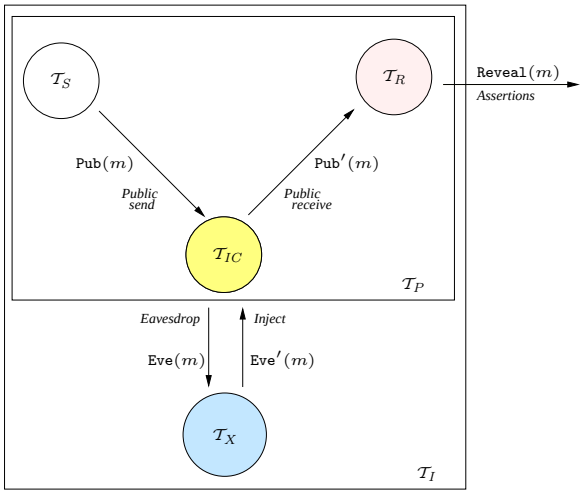

Figure 1. An example insecure scenario.

We let $\mathcal{T}_P$ denote the TA representing our protocol specification in the absence of the intruder. We thus define $\mathcal{T}_P$ to be the max-ai TA over $\{\mathcal{T}_S, \mathcal{T}_R, \mathcal{T}_{IC}\}$ that is obtained after hiding the actions $\Sigma^P_{com} = \Sigma^S_{com} \cup \Sigma^R_{com}$, *i.e.*, all messages sent through the insecure channel (*e.g.*, $\Sigma^P_{com} = \{\mathtt{Pub}, \mathtt{Pub}'\}$ in Fig. 1). Hence

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

Recall that, using the previously defined notations, the shortcut $\{\mathtt{Pub}, \mathtt{Pub}'\}$ stands for $\{\mathtt{Pub}(m), \mathtt{Pub}'(m) \mid \forall\, m \in \mathsf{Messages}\}$. We may use such shortcuts from now on when no confusion arises. By internalizing the actions Σ^P_{com}, these actions are no longer available for synchronizations in further TA composed over $\mathcal{T}_P$. To its environment, $\mathcal{T}_P$ thus appears as a black box, possibly with some output actions Σ^R_{sig}—signalling the successful reception of

messages. Usually such signals are used only for verification purposes (*e.g.*, $\Sigma^R_{sig} = \{\texttt{Reveal}\}$ in Fig. 1).

We let $\mathcal{T}_I$ denote the TA representing our protocol specification in the presence of the intruder. Actions Σ^I_{com} serve as the backdoor for intrusion and are added to $\mathcal{T}_{IC}$ (*e.g.*, $\Sigma^I_{com} = \{\texttt{Eve}, \texttt{Eve}'\}$ in Fig. 1). This is exactly what we need to guarantee that the intruder $\mathcal{T}_X$ may communicate with $\mathcal{T}_P$ only through the insecure channel. We thus define $\mathcal{T}_I$ to be the max-ai TA over $\{\mathcal{T}_P, \mathcal{T}_X\}$ that is obtained after hiding the actions Σ^I_{com}, *i.e.*, all messages that the intruder can eavesdrop from and inject back into the insecure channel. We thus enforce maximal synchronization between the intruder and the protocol. Hence

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

We have now defined an insecure communication scenario by composing a secure communication scenario with an intruder. Note that it would not be difficult to formally specify an insecure version of our TA model of the EMSS protocol. The extension considering an arbitrary number of receivers is also straightforward.

5. Security Analysis Through GNDC for TA

We now reformulate the GNDC schema in terms of TA. We assume some familiarity with process algebras.

Informally, the GNDC schema states that a system specification P satisfies property $GNDC^{\alpha(P)}_{\triangleleft}$ 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

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

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 required to be any process in the environment $\mathcal{E}_C$, *i.e.*, the set of all processes whose communicating actions are in C. By varying the parameters $\triangleleft$ and α, different security properties can be formulated [7, 10, 20].

In the specific context of analyzing cryptographic protocols, the *static* (initial) knowledge of the hostile environment must be bound to a specific set of messages. This limitation is needed to avoid a too strong hostile intruder that would be 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 actions C and having an initial knowledge of at most the messages in $\mathcal{D}(\phi)$. For the analysis of safety properties (*e.g.*, secrecy, integrity, and authentication) it is sufficient to consider the trace inclusion relation $\leq$ as behavioural relation between the terms of the algebra [9]. Hence, let us consider the GNDC instance

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

which was used to analyze integrity in stream signature protocols [10]. Informally, (1) requires traces of process $(P \parallel X)\backslash C$ (*i.e.*, the parallel composition of processes P and X restricted to communication over channels other than C) to be included in the traces of process $\alpha(P)$, representing the expected behaviour of P when no adversary is present.

5.1. Reformulating GNDC in Terms of TA

We begin by instantiating P to be a TA modelling a communication between a sender and a set of receivers through the use of an insecure channel, in the style of the TA $\mathcal{T}_P$ considered in the insecure scenario in Sect. 4. To this aim, we let $\mathcal{T}_P$ be specified as $\mathcal{T}_P = \{Q, (\Sigma^P_{inp}, \Sigma^P_{out}, \Sigma^P_{int}), \delta, I\}$. Because (1) requires P to communicate with X through the channels contained in C, we require a set C of actions for which $C \cap \Sigma^P_{ext} \neq \varnothing$ and $C \cap \Sigma^P_{com} = \varnothing$. This resembles requiring $\mathcal{T}_P$ to be able to communicate with the hostile environment $\mathcal{T}_X$ only by executing actions in Σ^I_{com} (*e.g.*, $\{\texttt{Eve}, \texttt{Eve}'\}$ in Fig. 1). In the sequel we thus assume C to coincide exactly with Σ^I_{com}. We are now able to formalize the hostile environment $\mathcal{E}_C$ in terms of TA as

$$\mathcal{E}_C = \{(Q, (\Sigma_{inp}, \Sigma_{out}, \Sigma_{int}), \delta, I) \mid \Sigma_{ext} \subseteq C\}.$$

In addition, (1) requires the initial knowledge of the environment to be bound to a specified 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 $\mathcal{D}(\phi)$. In terms of TA this means that a CA in the environment, when considered as a standalone component, can only execute output actions belonging to $c(\mathcal{D}(\phi))$, where $c \in C$. This is formally defined by restricting its behaviour to those traces consisting of solely output actions since—at a more abstract level—these are the traces that it can produce without receiving any additional messages from outside, *i.e.*, by exploiting only its own knowledge. Let $Id^{\Gamma}(\mathbf{B}_{\mathcal{T}}) = \{\gamma \in \mathbf{B}_{\mathcal{T}} \mid \gamma \in \Gamma^*\}$, where Γ is a set of actions. Consequently, the *initial knowledge* of $\mathcal{T}$ is defined as $Id^{\Sigma^{\mathcal{T}}_{out}}(\mathbf{B}_{\mathcal{T}})$. The formal definition of the environment $\mathcal{E}^{\phi}_C$ in terms of TA thus becomes

$$\mathcal{E}^{\phi}_C = \{\mathcal{X} \in \mathcal{E}_C \mid Id^{\Sigma^{\mathcal{X}}_{out}}(\mathbf{B}_{\mathcal{X}}) \subseteq (C(\mathcal{D}(\phi)))^*\}.$$

Finally, recall that we are only interested in output actions that a TA can produce by exploiting only its own knowledge. Therefore, we hide those output actions involved in communications and define the observational behaviour (w.r.t. actions not in C) of the resulting TA as those traces consisting of solely external actions not contained in C. As a result we are able to reformulate (1) in terms of TA.

Definition 9 *Let $\mathcal{T} = (Q, (\Sigma_{inp}, \Sigma_{out}, \Sigma_{int}), \delta, I)$ be a TA over $\mathcal{S}$, let $\Sigma_{com} \subseteq \Sigma_{ext}$, and let $\mathcal{T}' = hide_{\Sigma_{com}}(\mathcal{T})$. Then the* observational behaviour *of $\mathcal{T}'$ w.r.t. actions not in C, denoted by $\mathbf{O}^C_{\mathcal{T}'}$, is defined as*

$$\mathbf{O}^C_{\mathcal{T}'} = Id^{\Sigma^{\mathcal{T}'}_{ext} - C}(pres_{\Sigma^{\mathcal{T}'}_{ext}}(\mathbf{B}_{\mathcal{T}'})).$$

Definition 10 *Let $\alpha(\mathcal{T}_P)$ be the expected (correct) behaviour of $\mathcal{T}_P$. Then*

$$\mathcal{T}_P \in GNDC^{\alpha(\mathcal{T}_P)}_{\subseteq} \text{ iff } \forall\, \mathcal{X} \in \mathcal{E}^{\phi}_C : \mathbf{O}^C_{hide_{C(|||\{\mathcal{T}_P, \mathcal{X}\})}} \subseteq \alpha(\mathcal{T}_P).$$

Informally, Def. 10 says that $\mathcal{T}_P$ (*i.e.*, a communication protocol in our insecure scenario) satisfies $GNDC^{\alpha(\mathcal{T}_P)}_{\subseteq}$ if and only if its observational behaviour, despite communicating with any intruder $\mathcal{X}$ through the actions C, is included in $\alpha(\mathcal{T}_P)$ (*i.e.*, the expected correct behaviour of the communication protocol specified by $\mathcal{T}_P$). A significative instance of α is, *e.g.*, $\alpha_{int}(\mathcal{T}_P) = \mathbf{O}^C_{\mathcal{T}_P}$, which will be used in Sect. 6 to express integrity. Additionally, Def. 10 requires the intruder to be any TA able to interact with $\mathcal{T}_P$ through the actions C and with an initial knowledge bound to $\mathcal{D}(\phi)$.

5.2. Analysis Strategies for TA

While allowing a uniform approach to defining security properties, Def. 10 does not provide us with effective strategies for security analysis of cryptographic communication protocols. The universal quantification over $\mathcal{E}^{\phi}_C$ causes serious problems when deciding whether $\mathcal{T}_P \in GNDC^{\alpha(\mathcal{T}_P)}_{\subseteq}$. However, the theory developed for GNDC in terms of process algebras inspires similar methodologies within TA.

The Most General Intruder. As a first analysis strategy we look for a static characterization of the intruder, which does not involve the universal quantification of Def. 10.

Lemma 1 *Let $\mathcal{T}$ be a TA and let $\mathcal{X}, \mathcal{X}' \in \mathcal{E}_C$. Then*

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

Proof. Let $a_1 \cdots a_n \in \mathbf{O}^C_{\mathrm{hide}_{C(|||\{\mathcal{T}, \mathcal{X}\})}}$ and let $\mathbf{B}^C_{\mathcal{X}} \subseteq \mathbf{B}^C_{\mathcal{X}}$. Since $\mathcal{X} \in \mathcal{E}_C$, $\Sigma^{\mathcal{X}}_{ext} \subseteq C$. Then by Def. 9, for all $i \in [n]$, $a_i \in \Sigma^{\mathcal{T}}_{ext} - C$. We now use that by definition also all prefixes of $a_1 \cdots a_n$ are included in $\mathbf{O}^C_{\mathrm{hide}_{C(|||\{\mathcal{T}, \mathcal{X}\})}}$ and show by induction that all prefixes of $a_1 \cdots a_n$ are also included in $\mathbf{O}^C_{\mathrm{hide}_{C(|||\{\mathcal{T}, \mathcal{X}'\})}}$. First consider a_1. By Def. 9, either $a_1 \in \mathbf{B}_{\mathrm{hide}_{C(|||\{\mathcal{T}, \mathcal{X}\})}}$ or $b_1 \cdots b_m a_1 \in \mathbf{B}_{\mathrm{hide}_{C(|||\{\mathcal{T}, \mathcal{X}\})}}$, for some $m \geq 1$ and where, for all $j \in [m]$, b_j is an internal action of $\mathrm{hide}_C(|||\{\mathcal{T}, \mathcal{X}\})$. In both cases, since $\mathbf{B}^C_{\mathcal{X}} \subseteq \mathbf{B}^C_{\mathcal{X}'}$ and $a_i \in \Sigma^{\mathcal{T}}_{ext} - C$, for all $i \in [n]$, it follows by Def. 9 that $a_1 \in \mathbf{O}^C_{\mathrm{hide}_{C(|||\{\mathcal{T}, \mathcal{X}'\})}}$.

Now assume $a_1 \cdots a_k \in \mathbf{O}^C_{\mathrm{hide}_{C(|||\{\mathcal{T}, \mathcal{X}'\})}}$, with $k < n$, and consider $a_1 \cdots a_{k+1}$. By arguments as above and the induction hypothesis, $a_1 \cdots a_{k+1} \in \mathbf{O}^C_{\mathrm{hide}_{C(|||\{\mathcal{T}, \mathcal{X}'\})}}$. □

As $\mathcal{E}^{\phi}_C \subseteq \mathcal{E}_C$, this lemma holds for $\mathcal{X}, \mathcal{X}' \in \mathcal{E}^{\phi}_C$ as well. Based on the approach of [9] we now define a CA Top^{ϕ}_C, representing the *most general intruder*, in an attempt to circumvent the universal quantification of Def. 10. Note that the set C of predicates the intruder uses to interact with the insecure channel can be partitioned into C_{inp} and C_{out}, *i.e.*, the channel names the intruder uses to retrieve messages from and to inject messages back into the insecure channel, resp. (*e.g.*, in Fig. 1, $C_{inp} = \{\mathsf{Eve}\}$ and $C_{out} = \{\mathsf{Eve}'\}$).

Top^{ϕ}_C

Actions	States
Inp: C_{inp}(Messages)	received $\subseteq 2^{C_{inp}(\mathsf{Messages})}$,
Out: C_{out}(Messages)	initially ϕ
Int: $\varnothing$	

Transitions	
$c(m) \in C_{inp}$(Messages)	$c(m) \in C_{out}$(Messages)
Eff: received := received $\cup \{m\}$	Pre: $m \in \mathcal{D}$(received)

The general way in which this intruder is defined directly implies that its behaviour includes that of any TA from $\mathcal{E}^{\phi}_C$.

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

Lemmata 1 and 2 directly imply the following result.

Theorem 2 *For all $\mathcal{X} \in \mathcal{E}^{\phi}_C$,*

$$\mathbf{O}^C_{hide_{C(|||\{\mathcal{T}_P, \mathcal{X}\})}} \subseteq \mathbf{O}^C_{hide_{C(|||\{\mathcal{T}_P, Top^{\phi}_C\})}}.$$

Together with Def. 10, this gives us the following result.

Corollary 1 *Let $\alpha(\mathcal{T}_P)$ be as in Def. 10. Then*

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

Compositional Results. We now describe some results for the insecure scenario of Sect. 4. To begin with, we let

$$\begin{aligned} \mathcal{T}_1 &= \mathrm{hide}_{\Sigma^P_{com}}(|||\{\mathcal{T}_S, \mathcal{T}_{IC}\}) \text{ and} \\ \mathcal{T}_2 &= \mathrm{hide}_{\Sigma^P_{com}}(|||\{\mathcal{T}_R, \mathcal{T}_{IC}\}). \end{aligned}$$

We then let $\mathcal{T}_P$ be the TA defined at the end of Sect. 4, *i.e.*, with $\Sigma^I_{com} = C$ added to $\mathcal{T}_{IC}$. Now $\mathcal{T}_P$ thus represents the scenario in which an initiator and a responder are connected by an insecure channel, but not yet connected to an intruder. If we add the the most general intruder, some general results can be proved. To this aim we let

$$\begin{aligned} \mathcal{T}'_1 &= \mathrm{hide}_C(|||\{\mathcal{T}_1, Top^{\phi}_C\}) \text{ and} \\ \mathcal{T}'_2 &= \mathrm{hide}_C(|||\{\mathcal{T}_2, Top^{\phi}_C\}). \end{aligned}$$

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

$$\mathbf{O}^C_{\mathit{hide}_C(|||\{|||\{\mathcal{T}_1,\mathcal{T}_2\},\mathit{Top}^\phi_C\})} = \underline{||}_{\{\Sigma^{\mathcal{T}_1},\Sigma^{\mathcal{T}'_2}\}} \{\mathbf{O}^C_{\mathcal{T}'_1}, \mathbf{O}^C_{\mathcal{T}'_2}\}.$$

Proof. From the way $\mathcal{T}_1$ is composed it follows that $\Sigma^{\mathcal{T}_1}_{ext} = \Sigma^I_{com}$. Consequently, the way that $\mathcal{T}'_1$ is composed implies that $\Sigma^{\mathcal{T}'_1}_{ext} = \varnothing$. Hence, by Def. 9, $\mathbf{O}^C_{\mathcal{T}'_1} = \varnothing$, which trivially implies that $\underline{||}_{\{\Sigma^{\mathcal{T}'_1},\Sigma^{\mathcal{T}'_2}\}} \{\mathbf{O}^C_{\mathcal{T}'_1}, \mathbf{O}^C_{\mathcal{T}'_2}\} = \mathbf{O}^C_{\mathcal{T}'_2}$. It now remains to prove that $\mathbf{O}^C_{\mathcal{T}'_2} = \mathbf{O}^C_{\mathcal{T}''}$, where $\mathcal{T}'' = \mathrm{hide}_C(|||\{|||\{\mathcal{T}_1,\mathcal{T}_2\}, \mathit{Top}^\phi_C\})$. This however follows directly from the fact that $\{m \mid \{c(m) \in \Sigma^S_{com}\} \subseteq \phi$. □

Before continuing, we observe the following property of full synchronized shuffles. Let Δ_i, with $i \in [4]$, be alphabets and let $L_i \subseteq \Delta_i^*$. Then $L_1 \subseteq L_2$ and $L_3 \subseteq L_4$ implies that $\underline{||}_{\{\Delta_1,\Delta_3\}} \{L_1, L_3\} \subseteq \underline{||}_{\{\Delta_2,\Delta_4\}} \{L_2, L_4\}$.

Theorem 3 *If* $\mathcal{T}_1 \in \mathit{GNDC}^{\mathbf{O}^C_{\mathcal{T}_1}}_{\subseteq}$ *and* $\mathcal{T}_2 \in \mathit{GNDC}^{\mathbf{O}^C_{\mathcal{T}_2}}_{\subseteq}$, *then* $|||\{\mathcal{T}_1,\mathcal{T}_2\} \in \mathit{GNDC}^{\underline{||}_{\{\Sigma^{\mathcal{T}_1},\Sigma^{\mathcal{T}_2}\}}\{\mathbf{O}^C_{\mathcal{T}_1},\mathbf{O}^C_{\mathcal{T}_2}\}}_{\subseteq}$

Proof. We know from the hypothesis that $\mathbf{O}^C_{\mathcal{T}'_1} \subseteq \mathbf{O}^C_{\mathcal{T}_1}$ and $\mathbf{O}^C_{\mathcal{T}'_2} \subseteq \mathbf{O}^C_{\mathcal{T}_2}$. Hence, by Lemma 3 and the above observation, it follows that $\mathbf{O}^C_{\mathrm{hide}_C(|||\{|||\{\mathcal{T}_1,\mathcal{T}_2\},\mathit{Top}^\phi_C\})} = \underline{||}_{\{\Sigma^{\mathcal{T}'_1},\Sigma^{\mathcal{T}'_2}\}} \{\mathbf{O}^C_{\mathcal{T}'_1}, \mathbf{O}^C_{\mathcal{T}'_2}\} \subseteq \underline{||}_{\{\Sigma^{\mathcal{T}_1},\Sigma^{\mathcal{T}_2}\}} \{\mathbf{O}^C_{\mathcal{T}_1}, \mathbf{O}^C_{\mathcal{T}_2}\}$. □

6. Analysis of the EMSS Protocol

In this section we use the GNDC analysis framework for TA to show that integrity is guaranteed in an instance of the EMSS protocol. This has already been validated in [20], where a CCS-like process algebra was used. Here we deal with just one sender and one receiver and show the effectiveness of TA for security analysis.

We let $\mathcal{T}_S$ and $\mathcal{T}_R$ (incl. their occurrences in $\mathcal{T}_P$, etc.) be as specified in Subsection 3.1. Following the notation introduced at the beginning of Sect. 4, their actions shall henceforth be written as composed terms, *e.g.*, $\mathtt{Pub}(P_1)$'s outermost part Pub is a predicate symbol while its innermost part $P_1 = \langle m'_1, h(P_0), \varnothing\rangle$ is an atomic formula.

We formally 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 $\{\mathtt{Reveal}\}$. We require the expected (correct) observational behaviour $\alpha_{\mathit{int}}(\mathcal{T}_P)$ of $\mathcal{T}_P$ w.r.t. integrity as $\mathbf{O}^C_{\mathcal{T}_P}$, *i.e.*, as all prefixes of $\mathtt{Reveal}(m_0)\mathtt{Reveal}(m_1)\cdots\mathtt{Reveal}(m_{\mathit{last}})$.

Furthermore, we equip Top^ϕ_C with an initial knowledge ϕ consisting of all output actions of $\mathcal{T}_S$ as well as the public key $pk(\mathcal{T}_S)$, *i.e.*, $\phi = \{P_0, P_1, P_i, P_{\mathit{sign}} \mid 2 \le i \le \mathit{last}\} \cup \{pk(\mathcal{T}_S)\}$, 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 $1 \le i \le \mathit{last}$, and $P_{\mathit{sign}} = \langle\{h(P_{\mathit{last}}), h(P_{\mathit{last}-1})\}_{sk(\mathcal{T}_S)}\rangle$. We do so solely for analysis reasons, viz. in order to enable Top^ϕ_C to send the correct messages to $\mathcal{T}_R$ through the insecure channel when analyzing the max-ai TA over $\mathcal{T}_2$ and Top^ϕ_C (cf. the proof of Prop. 2). Note that the messages contained in its initial knowledge are exactly those that it is anyway able to receive in the max-ai TA over $\mathcal{T}_P$ and Top^ϕ_C by eavesdropping when $\mathcal{T}_S$ sends them through the insecure channel. As is common in security analysis, 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. Hence the observational behaviour of the max-ai TA over $\mathcal{T}_1$ and Top^ϕ_C is empty, *i.e.*,

Proposition 1 $\mathcal{T}_1 \in \mathit{GNDC}^{\varnothing}_{\subseteq}$.

Proof. Directly by Cor. 1, as $\mathbf{O}^C_{\mathrm{hide}_C(|||\{\mathcal{T}_1,\mathit{Top}^\phi_C\})} = \varnothing$. □

The way in which the receiver verifies the messages it receives implies that the observational behaviour of the max-ai TA over $\mathcal{T}_2$ and Top^ϕ_C is included in the expected observational behaviour $\alpha_{\mathit{int}}(\mathcal{T}_P)$ of $\mathcal{T}_P$ w.r.t. integrity, *i.e.*,

Proposition 2 $\mathcal{T}_2 \in \mathit{GNDC}^{\alpha_{\mathit{int}}(\mathcal{T}_P)}_{\subseteq}$.

Proof. The proof distinguishes two cases. If $\mathcal{T}_R$ does not output any action $\mathtt{Reveal}(m_i)$, for all payloads m_i, then its empty output behaviour is trivially included in $\alpha_{\mathit{int}}(\mathcal{T}_P)$.

Next assume that $\mathcal{T}_R$ outputs actions $\mathtt{Reveal}(m_i)$, for all payloads m_i. To do so, the definition of the EMSS protocol guarantees 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 TA ever outputs this private key, it follows that Top^ϕ_C does not know $sk(\mathcal{T}_S)$. Since digital signatures and hash functions cannot be forget by the intruder, the only possibility for $\mathcal{T}_R$ to output the $\mathtt{Reveal}(m_i)$, for all payloads m_i, is that Top^ϕ_C has sent $\mathcal{T}_R$ all the correct packages in the correct order. 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 $\mathcal{T}_S$. Hence $\mathcal{T}_R$ must output all payloads m_i in the correct order and thus $\mathbf{O}^C_{\mathrm{hide}_C(|||\{\mathcal{T}_2,\mathit{Top}^\phi_C\})} \subseteq \alpha_{\mathit{int}}(\mathcal{T}_P)$ because the behaviour of any TA is prefix closed. □

Finally, we can show that integrity is guaranteed in the instance of the EMSS protocol under scrutiny.

Proposition 3 $\mathcal{T}_P \in \mathit{GNDC}^{\alpha_{\mathit{int}}(\mathcal{T}_P)}_{\subseteq}$.

Proof. By Prop. 1 and 2 and Thm. 3, $|||\{\mathcal{T}_1,\mathcal{T}_2\} \in \mathit{GNDC}^{\underline{||}_{\{\Sigma^{\mathcal{T}_1},\Sigma^{\mathcal{T}_2}\}}\{\mathbf{O}^C_{\mathcal{T}_1},\mathbf{O}^C_{\mathcal{T}_2}\}}_{\subseteq} = \mathit{GNDC}^{\underline{||}_{\{\mathtt{Reveal}\}}\{\varnothing,\alpha_{\mathit{int}}(\mathcal{T}_P)\}}_{\subseteq} = \mathit{GNDC}^{\alpha_{\mathit{int}}(\mathcal{T}_P)}_{\subseteq}$. Then $\mathbf{O}^C_{\mathrm{hide}_C(|||\{|||\{\mathcal{T}_1,\mathcal{T}_2\},\mathit{Top}^\phi_C\})} \subseteq \alpha_{\mathit{int}}(\mathcal{T}_P)$ by Cor. 1. From Remarks 1 and 2 then follows that $\mathbf{B}_{|||\{|||\{\mathcal{T}_1,\mathcal{T}_2\},\mathit{Top}^\phi_C\}} = \mathbf{B}_{|||\{\mathcal{T}_S,\mathcal{T}_R,\mathcal{T}_{IC},\mathit{Top}^\phi_C\}} = \mathbf{B}_{|||\{\mathcal{T}_P,\mathit{Top}^\phi_C\}}$ and thus $\mathbf{O}^C_{\mathrm{hide}_C(|||\{|||\{\mathcal{T}_1,\mathcal{T}_2\},\mathit{Top}^\phi_C\})} =$

$\mathbf{O}^C_{\mathrm{hide}_{C(|||\{\mathcal{T}_P, Top^{\phi}_C\})}}$ by Def. 9. Hence $\mathbf{O}^C_{\mathrm{hide}_{C(|||\{\mathcal{T}_P, Top^{\phi}_C\})}}$ $\subseteq \alpha_{int}(\mathcal{T}_P)$ and thus, by Cor. 1, $\mathcal{T}_P \in GNDC^{\alpha_{int}(\mathcal{T}_P)}_{\subseteq}$. □

7. Conclusions and Future Work

We have embedded TA in a well-established framework for security analysis by reformulating the GNDC schema for TA. We have also achieved effective analysis strategies for a given insecure scenario with just two roles, viz. an initiator and a responder. First, by defining the most general intruder we have been able to avoid the universal quantification in an initial reformulation of the GNDC schema. Secondly, a compositional analysis strategy shows how security properties are preserved for a composition over these two roles. We conjecture that, on the basis of Thm. 1, these results can be extended to a multi-roled insecure scenario in which principals can act as initiators as well as responders. A detailed investigation into this direction is left as future work. We have then used the GNDC framework for TA to prove that integrity is guaranteed in a case study.

8. Acknowledgements

Ter Beek and Lenzini were supported by MIUR "Special Fund for the Development of Strategic Research" under CNR project "Instruments, Environments and Innovative Applications for the Information Society", subproject "Software Architecture for High Quality Services for Global Computing on Cooperative Wide Area Networks".

Lenzini was moreover partially supported by "PAW: Privacy in an Ambient World", a TUD/DIES/KUN/TNO-EIB/TNO-FEL collaboration funded by IOP GenCom under project nr. IGC03001.

Petrocchi was supported by MIUR Project: "Strumenti, Ambienti e Applicazioni Innovative per la Società dell'Informazione", sottoprogetto SP1: Reti INTERNET: "efficienza, integrazione e sicurezza".

References

[1] E. Badouel, Ph. Darondeau, D. Quichaud, and A. Tokmakoff. Modelling Dynamic Agent Systems with Cooperating Automata. Technical report, Publication Interne 1253, Institut de Recherche en Informatique et Systèmes Aléatoires, Rennes, 1999.

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

[3] M.H. ter Beek, C.A. Ellis, J. Kleijn, and G. Rozenberg. Synchronizations in Team Automata for Groupware Systems. *Computer Supported Cooperative Work—The Journal of Collaborative Computing*, 12(1):21–69, 2003.

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

[5] M.H. ter Beek, G. Lenzini, and M. Petrocchi. Team Automata for Security Analysis of Multicast/Broadcast Communication. In *Proc. WISP'03—affiliated to ATPN'03*, pages 57–71. Techn. Report Eindhoven Univ. of Technology, 2003.

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

[7] R. Focardi and R. Gorrieri. A Classification of Security Properties. *Journal of Computer Security*, 3(1), 1995.

[8] R. Focardi, R. Gorrieri, and F. Martinelli. Non-Interference for the Analysis of Cryptographic Protocols. In *Proc. ICALP'00*, LNCS 1853, pages 354–372. Springer, 2000.

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

[10] R. Gorrieri, F. Martinelli, M. Petrocchi, and A. Vaccarelli. Compositional Verification of Integrity for Digital Stream Signature Protocols. In *Proc. ACSD'03*. IEEE, 2003.

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

[12] R. Janicki and P.E. Laurer. *Specification and Analysis of Concurrent Systems, The COSY Approach*. Springer, 1992.

[13] N.W. Keesmaat. *Vector Controlled Concurrent Systems*. PhD thesis, Dept. of Computer Science, Leiden Univ., 1996.

[14] N.W. Keesmaat, H.C.M. Kleijn, and G. Rozenberg. Vector Controlled Concurrent Systems, Part I: Basic Classes. *Fundamenta Informaticae*, 13:275–316, 1990.

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

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

[17] G. Lenzini, S. Gnesi, and D. Latella. Spider: a Security Model Checker. In *Proc. FAST'03—affiliated to FME'03*, pages 163–180. Techn. Report ITT-CNR-10/2003, 2003.

[18] N.A. Lynch. I/O Automaton Models and Proofs for Shared-Key Communication Systems. In *Proc. CSFW-12*, pages 14–31, 1999.

[19] N.A. Lynch and M.R. Tuttle. An Introduction to Input/Output Automata. *CWI Quarterly*, 2(3):219–246, 1989.

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

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

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