Team Automata for Security Analysis of Multicast/Broadcast Communication*

Maurice ter Beek[1], Gabriele Lenzini[1], and Marinella Petrocchi[2]

[1] Istituto di Scienza e Tecnologie dell'Informazione, ISTI-CNR
Area della Ricerca di Pisa, Via G. Moruzzi 1, 56124 Pisa, Italy
{maurice.terbeek,gabriele.lenzini}@isti.cnr.it
[2] Istituto di Informatica e Telematica, IIT-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 to model secure multicast/broadcast communication with possible packet loss. This is a consequence of the natural way in which one-to-many (one-to-all) transmissions typical of multicast (broadcast) sessions can be modelled as communications between the component automata (CA) constituting a TA. To this aim, we use TA to model an instance of the EMSS multicast protocol family. In addition, we investigate a formulation of the Generalized Non-Deducibility on Compositions (GNDC) schema in terms of TA with the aim to embed TA in this well-established analysis framework. We intend to use this new setting for the formal verification of security properties for stream signature protocols.
Keywords:team automata, multicast/broadcast communication, GNDC

1 Introduction

Multicast/broadcast communication technology was born with the intent of saving bandwidth and CPU time with respect to the standard point-to-point connection known as unicast. A single virtual connection indeed uses no more bandwidth and resources for thousands of users than it does for a single user. Multicast and broadcast communication however present substantial differences. A sender transmitting a stream of data to a set of receivers could broadcast the stream to all the connected recipients (e.g. radio and TV broadcasts) or multicast the stream only to designated recipients (e.g. pay-per-view TV and conference calls). Multicast and broadcast data packets are usually sent over unreliable transport protocols, such as the User Data Protocol (UDP). This may cause packet loss (the stream is received incomplete by a part of the recipients). A variety of stream signature protocols dealing with the problem of authenticating streamed data over channels with packet loss has recently been proposed [10, 18, 19].

* ter Beek is supported by an ERCIM postdoctoral fellowship, Lenzini is supported by MIUR project SP4, and Petrocchi is partially supported by MURST Progetto "Metodi Formali per la Sicurezza ed il Tempo" (MEFISTO), by CNR project "Tecniche e Strumenti Software per l'Analisi della Sicurezza delle Comunicazioni in Applicazioni Telematiche di Interesse Economico e Sociale", and by a CSP grant for the project "SeTAPS II".

The aim of this paper is twofold. First it provides an initial investigation of the use of team automata (TA) for modelling stream signature protocols for secure multicast/broadcast communication in scenarios where packet loss may occur. To this aim an instance of one such a protocol, viz. the deterministic (1,2) schema of the Efficient Multi-chained Stream Signature (EMSS) protocol [19] is modelled by TA. Secondly, it inquires the possibility of defining a formal verification framework in terms of TA for analyzing the security of multicast/broadcast protocols. To this aim we study how the Generalized Non-Deducibility on Compositions (GNDC) schema may be reformulated in terms of TA, since this would embed TA in this well-established theory for defining and verifying a variety of security properties [7, 8, 17]. Generally speaking, a system specification P satisfies $GNDC_{\triangleleft}^{\alpha(P)}$ if and only if P, despite the fact that it is running in a hostile environment, appears indistinguishable from $\alpha(P)$ (w.r.t. a notion $\triangleleft$ of external observation). This $\alpha(P)$ represents the correct behaviour of P. Security properties such as NDC (Non-Deducibility on Compositions) and BNDC (Bisimulation-based NDC) [6] have been subsumed to GNDC instances [8]. Originally, the GNDC schema was formalized for a CCS-like process algebra and efficient verification strategies may be applied for verifying that P satisfies $GNDC_{\triangleleft}^{\alpha(P)}$ when specific conditions are met. Here we intend to take some preliminary steps towards formulating a TA version of GNDC.

TA form a flexible framework for modelling communication between system components [1, 2, 3], originally introduced in [5]. 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. TA model the logical architecture of a system by describing it solely in terms of an automaton, the role of actions and synchronizations between these actions. The crux of composing a TA is to define the way in which its constituting CA communicate by synchronizations reflecting the specific protocol of communication to be modelled. TA are an extension of input/output automata (IOA) [16, 22]. We now discuss this relation in an informal way. Formal comparisons can be found in [1, 2, 3]. The only difference between CA and IOA is the fact that the latter are input enabled: in each state it must be possible to execute every input action. Given a set $\mathcal{S}$ of CA, there is no such thing as the unique TA composed over $\mathcal{S}$. Rather, a whole range of TA, distinguishable only by their synchronizations, can be composed over this set of CA. IOA, on the other hand, are constructed according to a single and very strict method of composing automata, in effect resulting in composite automata that are uniquely defined by their constituents. Finally, IOA do not allow synchronizations of output actions, whereas TA do.

The fact that TA allow one-to-many types of synchronization between a set of CA makes them naturally suited for modelling multicast/broadcast communication. The basic communication mechanism in most CCS-like process algebras, however, is pairwise synchronization (in the form of an input/output handshake) between just two processes. This explains why in [17], where a CCS-like process algebra is used in order to exploit a well-established analysis framework, replication of pairwise synchronizations is used to simulate multicast/broadcast com-

munication. Moreover, packet loss is modelled by considering a receiver process that nondeterministically chooses whether or not to receive a packet. In the TA framework, also packet loss can be modelled in a natural way by varying the one-to-many type of synchronization per action. It would be interesting to further investigate the possibility to exploit process algebras like CSP [12], which can naturally express one-to-many communication, or the Calculus of Broadcasting Systems [20], which has broadcast as its main communication mechanism.

TA are compositions of CA that cooperate by synchronizing on certain actions. These synchronizations are labelled transitions describing state changes caused by global team actions. As a consequence, 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 have available the local information from which we can immediately extract which CA participate in a team synchronization. This visualizes the potential concurrency within a TA and thus relates TA to the world of Petri nets. In fact, this variant of TA called vector TA can be translated to a particular model of vector-labelled Petri nets from the framework of Vector Controlled Concurrent Systems (VCCSs), viz. Individual Token Net Controllers (ITNCs) [13, 14]. Though related, a number of important 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. Finally, ITNCs are finite-state systems, whereas (vector) TA may have an infinite number of states.

The paper is organized as follows. Section 2 lists some mathematical notations used throughout the paper and formally defines the framework of TA with multicast/broadcast communication. In Section 3 we briefly describe the EMSS protocol and show how to model its deterministic (1,2) schema by TA. In Section 4 we venture into the use of TA for security analysis by reformulating an instance of the GNDC schema in terms of TA. Section 5 concludes the paper.

2 TA with Multicast/Broadcast Communication

We begin this section with a list of mathematical notations that are used throughout this paper and which may be skipped on first reading. Consequently, we formally define the framework of TA with multicast/broadcast communication.

2.1 Some Notation

We assume some familiarity with basic notions of formal language theory [21].

Set inclusion is denoted by $\subseteq$. Set difference of sets V and W is denoted by $V - W$. The powerset of set V, formed by its finite parts only, is denoted by 2^V.

$\mathbb{N}$ denotes the set of positive integers. Let $\mathcal{I} \subseteq \mathbb{N}$ be a set of indices given by $\mathcal{I} = \{i_1, i_2, \dots\}$ with $i_j < i_\ell$ if $1 \leq j < \ell$. For sets V_i, $i \in \mathcal{I}$, we denote by $\prod_{i \in \mathcal{I}} V_i$

the (cartesian) product $\{(v_{i_1}, v_{i_2}, \dots) \mid v_{i_j} \in V_{i_j} \text{ for all } j \geq 1\}$. In addition to the prefix notation $\prod_{i\in\mathcal{I}} V_i$, we also use the infix notation $V_{i_1} \times V_{i_2} \times \cdots$. Let $j \in \mathcal{I}$. Then $\text{proj}_{\mathcal{I},j} : \prod_{i\in\mathcal{I}} V_i \to V_j$ is the function defined by $\text{proj}_{\mathcal{I},j}((a_{i_1}, a_{i_2}, \dots)) = a_j$. We thus note that if $\mathcal{I} = \{2, 3\}$, then $\text{proj}_{\mathcal{I},2}((a, b)) = a$.

Let $f : A \to A'$ and let $g : B \to B'$ be (total) 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 will use $f^{[2]}$ as shorthand notation for $f \times f : A \times A \to A' \times A'$. Thus $f^{[2]}(a, b) = (f(a), f(b))$. This notation should not be confused with iterated function application. In particular, we will use $\text{proj}_i{}^{[2]}$ as shorthand notation for $\text{proj}_i \times \text{proj}_i$. If $C \subseteq A$, then $f(C) = \{f(a) \mid a \in C\}$, and if $D \subseteq A \times A$, then $f^{[2]}(D) = \{(f(d_1), f(d_2)) \mid (d_1, d_2) \in D\}$.

Let Σ and Γ be two sets of symbols. Then the homomorphism $\text{pres}_{\Sigma,\Gamma} : \Sigma \to \Gamma^*$, 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 $\text{pres}_{\Sigma,\Gamma}$.

2.2 Team Automata (TA)

In this subsection we recall some definitions and results concerning TA from [3].

Definition 1. *An* automaton *is a construct $\mathcal{A} = (Q, \Sigma, \delta, I)$, with (possibly infinite) 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 the sequences $\alpha = q_0 a_1 q_1 a_2 q_2 \cdots$, with $q_0 \in I$ and, for all $i \geq 1$, $q_i \in Q$, $a_i \in \Sigma$, and $(q_{i-1}, a_i, q_i) \in \delta$, together with all their prefixes.*

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

The Σ-behaviour of $\mathcal{A}$ is also called the behaviour of $\mathcal{A}$. Let $a \in \Sigma$. The set δ_a of *a-transitions* of $\mathcal{A}$ is defined as $\delta_a = \{(q, q') \mid (q, a, q') \in \delta\}$ and a is *enabled* in $\mathcal{A}$ at state q, denoted by $a \,\text{en}_\mathcal{A}\, q$, if there exists a state q' of $\mathcal{A}$ such that $(q, q') \in \delta_a$.

A CA is an automaton distinguishing between output, input, and internal actions. Its internal actions have strictly local visibility and cannot be used for communication with other CA. Its 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 $\mathcal{C} = (Q, (\Sigma_{out}, \Sigma_{inp}, \Sigma_{int}), \delta, I)$, with underlying automaton $(Q, \Sigma_{out} \cup \Sigma_{inp} \cup \Sigma_{int}, \delta, I)$ and pairwise disjoint sets Σ_{out} of output, Σ_{inp} of input, and Σ_{int} of internal actions.* □

Σ denotes the set $\Sigma_{out} \cup \Sigma_{inp} \cup \Sigma_{int}$ of actions of $\mathcal{C}$ and Σ_{ext} denotes its set $\Sigma_{out} \cup \Sigma_{inp}$ of external actions.

For the rest of this section we let $\mathcal{S} = \{\mathcal{C}_i \mid i \in \mathcal{I}\}$ be an arbitrary but fixed set of CA specified as $\mathcal{C}_i = (Q_i, (\Sigma_{i,out}, \Sigma_{i,inp}, \Sigma_{i,int}), \delta_i, I_i)$, with set of actions $\Sigma_i = \Sigma_{i,out} \cup \Sigma_{i,inp} \cup \Sigma_{i,int}$.

When composing a TA over $\mathcal{S}$, we require the internal actions of the CA in $\mathcal{S}$ to be private, i.e. for all $i \in \mathcal{I}$, $\Sigma_{i,int} \cap \bigcup_{j\in(\mathcal{I}-\{i\})} \Sigma_j = \varnothing$. Such an $\mathcal{S}$ is called a *composable system*. For the rest of this section 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 for one or more of the CA becomes an output 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. Every internal action of the CA becomes an internal action 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 $a \in \bigcup_{i\in\mathcal{I}} \Sigma_i$. The set $\Delta_a(\mathcal{S})$ of* synchronizations *of a is defined as $\Delta_a(\mathcal{S}) = \{(q,q') \in \prod_{i\in\mathcal{I}} Q_i \times \prod_{i\in\mathcal{I}} Q_i \mid [\,\exists j \in \mathcal{I} : proj_j{}^{[2]}(q,q') \in \delta_{j,a}\,] \wedge [\,\forall i \in \mathcal{I} : [\,proj_i{}^{[2]}(q,q') \in \delta_{i,a}\,] \vee [\,proj_i(q) = proj_i(q')\,]\,]\}$.* □

$\Delta_a(\mathcal{S})$ 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 transformation of a state 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 $\mathcal{T} = (Q, (\Sigma_{out}, \Sigma_{inp}, \Sigma_{int}), \delta, I)$, with $Q = \prod_{i\in\mathcal{I}} Q_i$, $\Sigma_{out} = \bigcup_{i\in\mathcal{I}} \Sigma_{i,out}$, $\Sigma_{inp} = (\bigcup_{i\in\mathcal{I}} \Sigma_{i,inp}) - \Sigma_{out}$, $\Sigma_{int} = \bigcup_{i\in\mathcal{I}} \Sigma_{i,int}$, $\delta \subseteq Q \times \Sigma \times Q$, where $\Sigma = \Sigma_{out} \cup \Sigma_{inp} \cup \Sigma_{int}$, is such that $\{(q,q') \mid (q,a,q') \in \delta\} \subseteq \Delta_a(\mathcal{S})$, for all $a \in \Sigma$, and $\{(q,q') \mid (q,a,q') \in \delta\} = \Delta_a(\mathcal{S})$, for all $a \in \Sigma_{int}$, and $I = \prod_{i\in\mathcal{I}} I_i$.* □

Each choice of synchronizations thus defines a TA. At this point 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. 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.

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. Often the external actions to be hidden are those that are used in communications between CA.

Definition 6. *The set Σ_{com} of* communicating *actions of $\mathcal{S}$ is defined as $\Sigma_{com} = \{a \in \bigcup_{i\in\mathcal{I}} \Sigma_{i,ext} \mid \exists\, i,j \in \mathcal{I} : a \in (\Sigma_{i,inp} \cap \Sigma_{j,out}) \cup (\Sigma_{j,inp} \cap \Sigma_{i,out})\}$.* □

Secondly, it must be possible to construct unique TA of a certain specified type. This is the subject of the next subsection.

2.3 Multicast/Broadcast Communication in TA

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 these predicates is unique. Here we introduce some new predicates specifically for modelling multicast/broadcast communication in TA.

Definition 7. *Let $a \in \bigcup_{i\in\mathcal{I}} \Sigma_i$ and let $J \subseteq \mathcal{I}$. The predicate* J-cast *for a in $\mathcal{S}$, denoted by $\mathcal{R}_a^J(\mathcal{S})$, is defined as $\mathcal{R}_a^J(\mathcal{S}) = \{(q,q') \in \Delta_a(\mathcal{S}) \mid [\forall j \in J : [\, a \in \Sigma_j \wedge a\ en_{\mathcal{C}_j}\ proj_j(q))] \Rightarrow proj_j{}^{[2]}(q,q') \in \delta_{j,a}] \wedge [\forall\, i \in \mathcal{I}\setminus J : [proj_i(q) = proj_i(q')]\}$.* □

The predicate J-cast thus contains *all and only* those a-transitions from $\Delta_a(\mathcal{S})$ in which every CA from J participates *whenever* a is currently enabled in that CA, while none of the other CA does. These predicates thus model multicast communication between CA. Obviously, the $\mathcal{I}$-cast predicate models broadcast communication between CA. Hence in the sequel we may also refer to it as the *broadcast* predicate.[1]

Definition 8. *Let $J \subseteq \mathcal{I}$, let $\mathcal{R}_a^J(\mathcal{S}) \subseteq \Delta_a(\mathcal{S})$ for all $a \in \bigcup_{i\in\mathcal{I}} \Sigma_i$, and let $\mathcal{R}^J = \{\mathcal{R}_a^J(\mathcal{S}) \mid a \in \bigcup_{i\in\mathcal{I}} \Sigma_i\}$. Then $\mathcal{T} = (Q, (\Sigma_{out}, \Sigma_{inp}, \Sigma_{int}), \delta, I)$ is* the $\mathcal{R}^J$-TA *over $\mathcal{S}$, denoted by $||^J\mathcal{S}$, if $\delta_a = \mathcal{R}_a^J(\mathcal{S})$, for all $a \in \bigcup_{i\in\mathcal{I}} \Sigma_i$.* □

Each $\mathcal{R}^J$-TA over $\mathcal{S}$, with $J \subseteq \mathcal{I}$, is also called *the J-cast TA* over $\mathcal{S}$ because it is the unique TA with the following property: the addition of any synchronization results in a TA that contains a synchronization in which a CA from J in which a is currently enabled does not participate. Furthermore, the $\mathcal{R}^{\mathcal{I}}$-TA over $\mathcal{S}$ is also called *the broadcast TA* over $\mathcal{S}$ and it may also be denoted by $|||\mathcal{S}$.

3 A Case Study: The EMSS Protocol Modelled by TA

In this section we show how the TA framework can be used to model protocols for secure multicast/broadcast communication with possible packet loss. We do this by modelling a particular instance of the EMSS protocol in terms of TA.

3.1 The EMSS Protocol

The Efficient Multi-chained Stream Signature (EMSS) protocol to sign digital streams was introduced in [19]. EMSS exploits a combination of hash functions and digital signatures and—contrary to previous proposals [9]—achieves (some) robustness against packet loss. A stream signature protocol tolerates packet loss

[1] The broadcast predicate is called *is-state-indispensable* in [3].

whenever 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. Here we focus on a specific instance of this family, viz. the deterministic (1,2) schema. We assume that a sender S wants to send a stream of messages $m_0, m_1, \ldots, m_{last}$ to a set of receivers $\{R_n \mid n \geq 1\}$.[2] The protocol then requires S to send triples of messages (called packets) to the receivers. After an initial phase, each packet P_i contains a meaningful payload m_i,[3] 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.[4] The protocol can formally be described as follows.

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

3.2 The EMSS Protocol Modelled by TA

In this subsection we use the TA framework to model the deterministic (1,2) schema of the EMSS family of protocols. 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 of a CA $\mathcal{T}_R$. $\mathcal{T}_S$ uses its private key $sk(\mathcal{T}_S)$ and a public key $pk(\mathcal{T}_S)$ to perform regular digital signature operations. Let $\mathsf{Messages}$ denote the set $\{m_0, m_1, \ldots, m_{last}\}$ of meaningful payloads. Then $\mathcal{T}_S$ uses the hash function $h : \mathsf{Messages} \rightarrow \mathsf{Hashed}$, while $\mathcal{T}_R$ uses the hash function $\bar{h} = h$. Moreover, $\mathcal{T}_S$ uses the function $s : 2^{\mathsf{Hashed}} \rightarrow \mathsf{Signed}$, defined by $s(H) = H_{sk(\mathcal{T}_S)}$, to sign sets of hashed messages with its private key $sk(\mathcal{T}_S)$, whereas $\mathcal{T}_R$ uses the function $\bar{s} : \mathsf{Signed} \rightarrow \{\mathsf{true}, \mathsf{false}\}$ and the public key $pk(\mathcal{T}_S)$ to verify whether or not a set of hashed messages was signed by $\mathcal{T}_S$.

Before presenting the specification of $\mathcal{T}_S$ we note that we specify TA in the way IOA are commonly defined [15, 16]. The notation for their sets of states and transtitions might require some further explanation. The states of a TA are defined by the current values of the variables listed under States. The transitions of a TA are defined, per action a, as preconditions (Pre) and effect (Eff): (q, a, q') is a transition of $\mathcal{T}_S$ whenever the precondition of a is satisfied by q and q' is the transformation of q defined by the effect of a. We omit the precondition for an action when this precondidition is true.

[2] As is usual for recipients of digital data streams, the receivers are not able to communicate among each other.

[3] We assume the private sender key $sk(S)$ cannot be deduced from $\{m_i \mid 0 \leq i \leq last\}$.

[4] In [19], loss tolerance is increased by sending multiple copies of the signature packet.

$\mathcal{T}_S$

Actions

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

Input: $\varnothing$

Internal: $\{Hash_i \mid 0 \leq i \leq last\} \cup \{Sign\}$

States

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

Transitions

P_0
Eff: $\mathsf{sent} := \mathsf{sent} \cup \{P_0\}$

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

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

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

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

P_{sign}
Pre: $\{h(P_{last}), h(P_{last-1})\}_{sk(\mathcal{T}_S)} \in \mathsf{signed}$
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 $P_0 P_1 \cdots P_{last} P_{sign}$. To actually send the packets $P_0, P_1, \ldots, P_{last}$, P_{sign} in this particular order, $\mathcal{T}_S$ needs to 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 $Hash_0 Hash_1 \cdots Hash_{last} Sign$.

We continue with the specification of $\mathcal{T}_R$, capable of receiving all packets $P_0, P_1, \ldots, P_{last}, P_{sign}$ as input behaviour. Upon receiving P_i, $\mathcal{T}_R$ verifies whether or not it has received P_{i-1}. First consider the case 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 verifiable payload m_{i-1} is extracted from P_{i-1}. If they are not equal, then $\mathcal{T}_R$ has no output behaviour.

Secondly, consider the case that $\mathcal{T}_R$ did not receive P_{i-1}. Then it verifies whether or not 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 or not 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 verifiable payload m_{i-2} is extracted from P_{i-2}. If they are not equal, then $\mathcal{T}_R$ has no output behaviour.

Eventually $\mathcal{T}_R$ receives the signature packet P_{sign},[5] after which it verifies the accompanying digital signature,[6] 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 $\mathcal{T}_R$.

$\mathcal{T}_R$

Actions

Output: Messages
Input: $\{P_i \mid 0 \leq i \leq last\} \cup \{P_{sign}\}$
Internal: $\{XtractH_i, XtractM_i, Hash_i \mid 0 \leq i \leq last\} \cup \{Verify, Stream\}$

States

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

Transitions

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

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

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

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

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

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

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

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

[5] We assume P_{sign} is always received. However, 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.

[6] We assume $\mathcal{T}_R$ has previously retrieved the public key $pk(\mathcal{T}_S)$ corresponding to the private key $sk(\mathcal{T}_S)$.

$XtractM_i,\ 0 \leq i \leq last$
Pre: $[\, h(P_i) \in \mathsf{xtractedH}\,]$
$\wedge\, [\bar{h}(P_i) \in \mathsf{hashed}] \wedge [\bar{h}(P_i) = h(P_i)]$
Eff: $\mathsf{xtractedM} := \mathsf{xtractedM} \cup \{m_i\}$

m_0
Pre: $[\mathsf{send} = \mathsf{true}] \wedge [m_0 \in \mathsf{xtractedM}]$
Eff: $\mathsf{xtractedM} := \mathsf{xtractedM} - \{m_0\}$

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

$m_i,\ 1 \leq i \leq last$
Pre: $[\mathsf{send} = \mathsf{true}] \wedge [m_i \in \mathsf{xtractedM}]$
$\wedge\, [\{m_k \mid 0 \leq k < i\} \cap \mathsf{xtractedM} = \varnothing]$
Eff: $\mathsf{xtractedM} := \mathsf{xtractedM} - \{m_i\}$

Clearly the input behaviour $\mathbf{B}_{\mathcal{T}_R}^{\Sigma_{inp}}$ of $\mathcal{T}_R$ 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 $XtractH_{1,1} Hash_0 XtractM_0 XtractH_{2,1} Hash_1 XtractM_1 \cdots XtractH_{last,1}$ $Hash_{last-1} XtractM_{last-1} VerifyXtractH_{sign,1} Hash_{last} XtractM_{last} Stream$, as well as—reflecting a different order of performing the same series of internal computations—$XtractH_{1,1} XtractH_{2,1} \cdots XtractH_{last,1} Hash_0 XtractM_0 Hash_1 XtractM_1 \cdots$ $Hash_{last-1} XtractM_{last-1} VerifyXtractH_{sign,1} Hash_{last} XtractM_{last} Stream$ and—reflecting yet another different order—$XtractH_{1,1} XtractH_{2,1} \cdots XtractH_{last,1}$ $VerifyHash_0 XtractM_0 Hash_1 XtractM_1 \cdots Hash_{last-1} XtractM_{last-1} XtractH_{sign,1}$ $Hash_{last} XtractM_{last} Stream$. Finally, the output behaviour $\mathbf{B}_{\mathcal{T}_R}^{\Sigma_{out}}$ of $\mathcal{T}_R$ consists of all prefixes of $m_0 m_1 \cdots m_{last}$.

Let $\mathcal{C}_1 = \mathcal{T}_S$ and let $C_i = \mathcal{T}_R$,[7] for all $2 \leq i \leq n+1$. Then the $\{1\}$-cast TA over $\{\mathcal{C}_i \mid 1 \leq i \leq n+1\}$ is essentially the same as $\mathcal{T}_S$. Since it has the same output behaviour as $\mathcal{T}_S$, it thus models multicast/broadcast communication with full packet loss. The K-cast TA over $\{\mathcal{C}_i \mid 1 \leq i \leq n+1\}$, where $K \supset \{1\}$, has the union of the output (internal) actions of $\mathcal{T}_S$ and $\mathcal{T}_R$ as its output (internal) actions and it has no input 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 multicast/broadcast communication.

Finally, we illustrate how we can model multicast/broadcast communication with some packet loss by varying the type of synchronization per output action. Assume that $\mathcal{T}_S$ performs a multicast communication with two receivers, viz. the jth and the kth receiver, and let $L = \{1\} \cup \{j, k \mid 2 \leq j < k \leq n+1\}$. If there would be no packet loss, then we would compose the L-cast TA over $\{\mathcal{C}_i \mid 1 \leq i \leq n+1\}$. Let Σ be the alphabet of this TA. Then we recall that the L-cast TA may also be called the $\mathcal{R}^L$-TA or—even more detailed—the $\{\mathcal{R}_a^L \mid \forall\, a \in \Sigma\}$-TA over $\{\mathcal{C}_i \mid 1 \leq i \leq n+1\}$. Next we assume that there is some packet loss, viz. the jth receiver does not receive P_1 and the kth receiver does not receive P_{last-1}. To reflect this packet loss, we would compose the $(\{\mathcal{R}_a^L \mid \forall\, a \in (\Sigma - \{P_1, P_{last-1}\})\} \cup \{\mathcal{R}_{P_1}^{L-\{j\}}\} \cup \{\mathcal{R}_{P_{last-1}}^{L-\{k\}}\})$-TA over $\{\mathcal{C}_i \mid 1 \leq i \leq n+1\}$.

[7] Strictly spoken, the internal actions of each $\mathcal{T}_R$ must be indexed to satisfy the composability condition.

4 Towards Security Analysis with TA

In this section we perform an initial investigation of how TA can be used to analyze security of cryptographic protocols, in particular those based on multicast/broadcast communication. By doing this in a rather generic way, we hope to achieve that this framework can eventually serve as a general setup for the analysis and verification of a variety of security properties for cryptographic protocols of communication.

4.1 The Insecure Communication Scenario

Contrary to the direct multicast/broadcast communication between sender and receivers considered so far, we now assume that a sender $\mathcal{T}_S$ sends messages to a set of receivers $\mathcal{T}_R$ over an *insecure channel*. We moreover introduce an *intruder*, i.e. an eavesdropper, which is able to listen in on this insecure channel in such a way that it can modify (fake) all messages passing through this channel. When verifying security properties for communication protocols, it is quite common to include (sometimes implicitly) an additional intruder component which 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 (e.g. integrity) if it satisfies this property despite the presence of the intruder. Based on the approach of [15], 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:

(a) $\mathcal{T}_S$ plays the role of the protocol's honest sender,
(b) each $\mathcal{T}_R$ plays the role of one of the protocol's honest receivers,
(c) $\mathcal{T}_{IC}$ plays the role of an insecure channel, and
(d) $\mathcal{T}_X$ plays the role of the intruder or hostile environment.

We assume the actions of these TA to be built over a first order signature, where predicate symbols are seen as communication ports and atoms as messages. We abstract from the cryptographic details concerning 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 of messages ϕ, a new set of messages $\mathcal{D}(\phi)$ can be obtained, i.e. $\mathcal{D}(\phi) = \{m \mid \phi \vdash m\}$. This is a standard approach in the formal analysis of cryptographic protocols of communication [8, 15].

We do not explicitly model the TA of our framework, but we instead describe them informally by their interactions in Figure 1. When $\mathcal{T}_S$ attempts to send messages to $\mathcal{T}_R$ through the public (insecure) channel, then these messages may be eavesdropped by the intruder. The intruder is able to modify the messages it has eavesdropped and to inject them back into the public channel.

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 broadcast TA over $\{\mathcal{T}_S, \mathcal{T}_R, \ldots,$

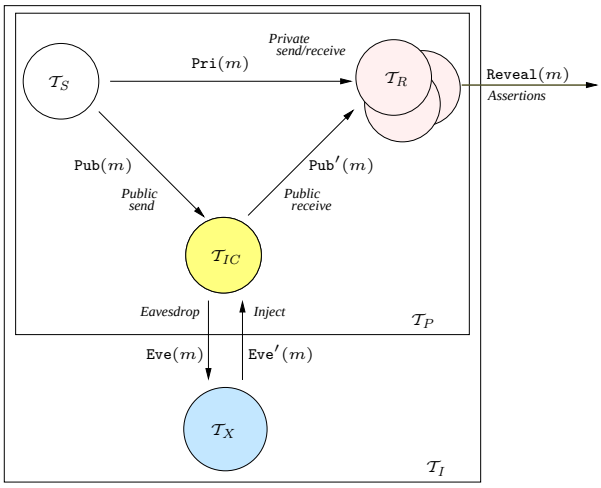

Fig. 1. Interactions between the TA involved in the insecure communication scenario.

$\mathcal{T}_R, \mathcal{T}_{IC}\}$ that is obtained after hiding the communicating actions $\Sigma^P_{com} = \{\mathtt{Pri}(m), \mathtt{Pub}(m), \mathtt{Pub}'(m) \mid \forall \text{ messages } m\}$ of $\{\mathcal{T}_S, \mathcal{T}_R, \ldots, \mathcal{T}_R, \mathcal{T}_{IC}\}$,[8] i.e. all the messages that are sent through either the public or the insecure channel. Hence

$$\mathcal{T}_P = hide_{\Sigma^P_{com}}(|||\{\mathcal{T}_S, \mathcal{T}_R, \ldots, \mathcal{T}_R, \mathcal{T}_{IC}\}). \tag{1}$$

Alternatively, if we want to model a multicast communication between the sender and a subset of the receivers, then the J-cast operator $\|^J$ should be used in (1). By internalizing the communicating 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 with output actions $\mathtt{Reveal}(m)$—signalling the successful reception of messages[9]—and $\mathtt{Eve}(m)$—serving as the back door for intrusion. 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 let $\mathcal{T}_I$ denote the TA representing our protocol specification in the presence of the intruder. We thus define $\mathcal{T}_I$ to be the broadcast TA over $\{\mathcal{T}_P, \mathcal{T}_X\}$ that is obtained after hiding the communicating actions $\Sigma^I_{com} = \{\mathtt{Eve}(m), \mathtt{Eve}'(m) \mid \forall \text{ messages } m\}$ of $\{\mathcal{T}_P, \mathcal{T}_X\}$, i.e. all messages that the intruder can eavesdrop from and inject into the insecure channel. Here the use of the broadcast operator just forces maximal synchronization between the intruder and the protocol. Hence

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

We have now defined an insecure communication scenario by composing a secure communication scenario with a TA specifying an intruder. Note that it is not difficult to formulate an insecure communication scenario for our TA model of the deterministic (1,2) schema of the EMSS protocol defined in Section 3.

[8] In the setting of [3], $\mathcal{T}_P$ would be called the *maximal-si (communication) closed* TA.
[9] Usually these signals are used only for verification purposes.

4.2 Generalized Non-Deducibility on Compositions for TA

In this section we reformulate an instance of the Generalized Non-Deducibility on Compositions (GNDC) schema, as originally defined in [8] in the context of a CCS-like process algebra, in terms of TA. This embeds TA in this general and well-established analysis framework for defining and verifying security properties. We assume some familiarity with the basic notions from process algebras [4].

Informally, the GNDC schema states that a system specification P satisfies property $GNDC_{\triangleleft}^{\alpha(P)}$ if the behaviour of P, despite the presence of a hostile environment $\mathcal{E}_{\mathcal{C}}^{\phi}$ that can interact with P only through a fixed set of channels C, *appears* to be 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

$$P \in GNDC_{\triangleleft}^{\alpha(P)} \text{ iff } \forall X \in \mathcal{E}_{C}^{\phi} : (P \parallel X)\backslash C \triangleleft \alpha(P), \tag{2}$$

where $(P \parallel X)\backslash C$ denotes the parallel composition of processes P and X restricted to communication over channels other than $\mathcal{C}$. By varying parameters $\triangleleft$ and α a variety of security properties can be formulated in the GNDC schema [6].

In (2) there is an additional constraint that is required 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 environment that would be able to corrupt any secret (as it would know all cryptographic keys, etc.). Formally, $\mathcal{E}_{C}^{\phi}$ just represents an environment with an initial knowledge bound to at most the messages in $\mathcal{D}(\phi)$. Obviously, with a specific formal framework in mind, all symbols in (2) need to be instantiated. As an example, consider the instance

$$P \in GNDC_{\leq_{trace}}^{P\backslash C} \text{ iff } \forall X \in \mathcal{E}_{C}^{\phi} : (P \parallel X)\backslash C \leq_{trace} P\backslash C \tag{3}$$

of GNDC, which was used in [11, 17] to analyze integrity properties in stream signature protocols. Informally, (3) 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 $\mathcal{C}$) to be included in the traces of process P in which communications over C are not allowed. This instance is thus a natural choice for initiating a reformulation in terms of TA.

We begin by instantiating P to be a TA modelling broadcast communication between a sender and a set of receivers through the use of an insecure public channel, in the style of the TA $\mathcal{T}_P$ considered before. We let $\mathcal{T}_P$ be specified as $\mathcal{T}_P = \{Q, (\Sigma_{out}^P, \Sigma_{inp}^P, \Sigma_{int}^P), \delta, I\}$. Because (3) requires P to communicate with X through the channels contained in C, we require $C \cap \Sigma_{out}^P \neq \varnothing$, $C \cap \Sigma_{inp}^P \neq \varnothing$, and $C \cap \Sigma_{com}^P = \varnothing$. This resembles requiring $\mathcal{T}_P$ to be able to communicate with the hostile environment $\mathcal{T}_X$ only by executing actions contained in $\{\mathtt{Eve}(m), \mathtt{Eve}'(m) \mid \forall \text{ messages } m\}$. We are now able to formalize the hostile environment $\mathcal{E}_C$ in terms of TA as

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

In addition, (3) requires the initial knowledge of the environment to be bound to a specified set of messages ϕ. Informally this means that the environment should be able to produce, by means of its internal functioning, at most the set of messages contained in $\mathcal{D}(\phi)$. In terms of TA this means that a TA in the environment, when considered as a stand-alone component, can only execute output actions that belong to $\mathcal{D}(\phi)$. This is formally defined by restricting its behaviour to those sequences consisting of solely output actions since—at a more abstract level—these are the sequences 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 $Id^{\Sigma_{out}}(\mathbf{B}_{\mathcal{T}})$. At this point the formal definition of the environment $\mathcal{E}_C^{\phi}$ in terms of TA is

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

Finally, we define the observational behaviour excluding actions from C of a TA as those sequences consisting of solely input and output actions that are not involved in any communication and that moreover are not contained in C. The observational behaviour plays the same role here as the set of traces does in [8].

Definition 9. *Let $\mathcal{T}$ be a TA composed over a set of at least two CA. The* observational behaviour *of $\mathcal{T}$, denoted by $\mathbf{O}_{\mathcal{T}}^C$, is defined as*

$$\mathbf{O}_{\mathcal{T}}^C = Id^{\Sigma_{ext}-C}(pres_{\Sigma_{ext}-\Sigma_{com}}(\mathbf{B}_{\mathcal{T}})). \qquad \square$$

We are now able to reformulate (3) in terms of TA.

Definition 10. *Let $\mathcal{T}$ be a TA composed over a set of at least two CA. Then*

$$\mathcal{T} \in GNDC_{\subseteq}^C \text{ iff } \forall \mathcal{X} \in \mathcal{E}_C^{\phi} : \mathbf{O}_{hide_C(|||\{\mathcal{T},\mathcal{X}\})}^C \subseteq \mathbf{O}_{\mathcal{T}}^C. \qquad \square$$

When used in the insecure scenario of the previous subsection and under the condition that receivers claim to have successfully received every message, this definition expresses integrity.

5 Conclusions and Future Work

This paper presents TA as a natural formal model for multicast/broadcast communication. In fact, in the theory of TA, many variants of a concurrent composition operator can be uniformly defined. In particular, we defined a multicast composition operator $\|^J$ so that we were able to model a multicast protocol involving one sender $\mathcal{T}_S$ and n copies of a receiver $\mathcal{T}_R$ as one-to-J synchronizations between the components of a TA. We showed the effectiveness of such a formalization by providing the specification of an instance of the Efficient Multi-chained Stream Signature (EMSS) protocol. In addition, in order to embed TA in a well-established analysis framework, we initiated a study of how to reformulate the GNDC schema for TA. As an example we dealt with expressing integrity properties as GNDC using TA. As future work we intend to study whether and how analysis techniques, as existing for GNDC, such as a static characterization of GNDC and compositionality proofs, can migrate in the world of TA.

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, and G. Rozenberg. Team Automata for CSCW. In *Proc. 2nd International Colloquium on Petri Net Technologies for Modelling Communication Based Systems*, pages 1–20, 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] J.A. Bergstra, A. Ponse, and S.A. Smolka, editors. *Handbook of Process Algebra*. Elsevier Science, 2001.

[5] C.A. Ellis. Team Automata for Groupware Systems. In *Proc. International ACM SIGGROUP Conference on Supporting Group Work: The Integration Challenge (GROUP'97)*, pages 415–424. ACM Press, 1997.

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

[7] R. Focardi, R. Gorrieri, and F. Martinelli. Non-Interference for the Analysis of Cryyptographic Protocols. In *Proc. ICALP'00*, LNCS 1853. Springer, 2000.

[8] 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.

[9] R. Gennaro and P. Rohatgi. How to Sign Digital Streams. *Information and Computation*, 165(1):100–116, 2001.

[10] P. Golle and N. Modadugu. Authenticating Streamed Data in the Presence of Random Packet Loss. In *Proc. NDSS'01*, 2001.

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

[12] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[13] N.W. Keesmaat. *Vector Controlled Concurrent Systems*. PhD thesis, Department of Computer Science, Leiden University, 1990.

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

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

[17] F. Martinelli, M. Petrocchi, and A. Vaccarelli. Analysing EMSS with Compositional Proof Rules for Non-Interference. In *Proc. WITS'03*, pages 52–61, 2003.

[18] A. Pannetrat and R. Molva. Efficient Multicast Packet Authentication. In *Proc. NDSS'03*, 2003.

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

[20] K.V.S. Prasad. A Calculus of Broadcasting Systems. In *Proc. 16th Colloquium on Trees in Algebra and Programming (TAPSOFT-CAAP 1991)*, LNCS 493, pages 338–358, 1991.

[21] A. Salomaa. *Formal Languages*. Academic Press, 1973.

[22] M.R. Tuttle. Hierarchical Correctness Proofs for Distributed Algorithms. Master's thesis, Department of Electrical Engineering and Computer Science, MIT, 1987.