SECCO 2004 Preliminary Version

Team Automata for Security – A Survey –

Maurice H. ter Beek[a] Gabriele Lenzini[b] Marinella Petrocchi[c]

[a] *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

[b] *Department of Computer Science, University of Twente*
P.O. Box 217, 7500 AE Enschede, The Netherlands
lenzinig@cs.utwente.nl

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

Abstract

In [33], Kleijn presented a survey of the use of team automata for the specification and analysis of phenomena from the field of computer supported cooperative work, in particular notions related to groupware systems. In this paper we present a survey of the use of team automata for the specification and analysis of some issues from the field of security. In particular, we show how team automata can adequately be used to model and verify various access control policies, multicast/broadcast communication protocols, and general (cryptographic) communication protocols.

Key words: team automata, access control, security, cryptographic communication protocols

1 Introduction

Team Automata (TA) have originally been introduced in the context of Computer Supported Cooperative Work (CSCW) [5,18], but they have since proved their use also in the context of security [3,4,6,7]. In [33] a survey of their use for CSCW was presented. In this paper we survey the use of TA for security.

TA are inspired by—and form an extension of—Input/Output automata (IOA) [40]. Like IOA, TA form a flexible framework for modelling communication between components of distributed and reactive systems. They 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. A TA is composed of component automata (CA), which are ordinary automata without final states and with a distinction of their actions into input, output and internal actions. The only difference between a CA and an IOA is that IOA are by definition *input enabled*: in each state it must be possible to execute every input action. The crux of composing a TA is to define the way in which its constituting CA communicate by synchronizations. Whereas IOA 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, there is no such thing as the unique TA composed over a given set of CA. Rather, a whole range of TA, distinguishable only by their synchronizations, can be composed over this set of CA. In particular, contrary to the case of IOA, in TA also output actions may be synchronized upon.

The rigorous setup of these frameworks allows one to formulate and verify general and specific logical properties of complex (distributed, reactive) systems in a mathematically precise way. In realistically large computer systems, security is a big issue, and these frameworks allow formal proofs of correctness of its design. Moreover, such a formal approach forces one to unambiguously describe one's design and it may suggest new approaches not seen otherwise. The particular characteristics of TA with respect to IOA were showed to be useful in specific circumstances, two of which we describe next. In [4], ter Beek *et al.* use the synchronization of output actions to define so-called *peer-to-peer* and *master-slave* synchronizations. These are two important CSCW phenomena, which were thus introduced with a clear practical motivation in mind. Neither of them can however be distinguished in IOA. In [6], ter Beek *et al.* use the freedom of choosing the synchronizations of a TA over a set of CA to define a so-called *multicast* composition operator $\|^J$ as a one-to-J synchronization between a sender and a subset J of the total set of receivers. This notion cannot be distinguished in IOA.

This paper is a tutorial overview of research conducted in the last four years on the use of TA for the specification and analysis of security issues. In particular, it covers the specification of several access control strategies by ter Beek *et al.* [4]—consequently verified by ter Beek and Bloem in [3]—as well as ongoing work by the authors of the present paper on developing a TA framework for the analysis of security properties—initiated in [6] and further developed in [7]. We now describe each of these approaches in more detail. [1]

In [4], ter Beek *et al.* demonstrate the potential of TA for capturing information security and protection structures, and critical coordinations between these structures. On the basis of a spatial access metaphor, various known access control strategies are formally specified in terms of synchronizations in TA. In [3], ter Beek and Bloem moreover initiated an attempt to validate some of the resulting specifications with the model checker Spin [32].

[1] Very recently, a first attempt to use TA for the analysis of privacy properties was conducted by Egidi and Petrocchi [16]. While their work is too recent for a full discussion in this paper, we note that the authors use TA to model and verify a protocol for securing agents in a hostile environment, focusing on privacy properties of the agents.

In [6], subsequently, the authors of the present paper showed the potential of TA for modelling secure multicast and broadcast communication. To this aim, TA are used to model an instance of a particular stream signature protocol. The one-to-many and one-to-all communications that are so typical of multicast and broadcast communications, are captured by TA in a native way as synchronizations between the set of CA constituting a TA.

In [7], finally, the author of the present paper develop a framework for security analysis with TA. First they define an insecure communication scenario for TA, based on the addition of a so-called *most general intruder* to a TA model of a secure communication protocol. The intruder is modelled as an active agent able to influence the communication among honest agents. This insecure scenario is general enough to encompass various communication protocols. Secondly, the Generalized Non-Deducibility on Compositions (GNDC) schema of [26] is reformulated in terms of TA and, subsequently, a compositional analysis strategy is described for it, which can be used for verifying security properties in the communication protocol modelled by the scenario. Thirdly, this framework is applied to show that integrity is guaranteed for a case study in which TA model a particular instance of the Efficient Multi-chained Stream Signature (EMSS) protocol family of [46]. This case study shows the effectiveness of this TA approach for a realistic stream signature protocol, thus facilitating an easy comparison for those familiar with other approaches.

The approach to use an automata-based formalism for the specification and verification of properties in the field of security is not unique, but has become very popular in recent years [11,12,29,35,38,44,45,47]. We briefly describe some approaches closest to those surveyed in this paper.

Ongoing work by Chen *et al.* on the formal verification of security properties of software is discussed in [11]. Their specific approach is to model the particular software to be verified as a pushdown automaton and the security property as a finite automaton, while model-checking techniques are used to actually perform the verification. An application of their approach can be found in [12], where the access control policies of three different Unix systems are described and analysed.

In [47], Schneider defines so-called *Security Automata* as a form of Büchi automata, similar to ordinary (finite) automata, and applies them to a simple access control model. Similar to composition of IOA and TA, so-called *conjunction security automata* are defined. It remains to be seen whether also complex access control policies with delegation and revocation can be modelled by security automata or in the approach of Chen *et al.*. We anticipate that in [4], ter Beek *et al.* use TA exactly for such policies.

An experiment involving the combination of simple shared-key communication with the Diffie-Hellman key distribution protocol [14] is modelled and proved correct using IOA by Lynch in [38]. As noted by the author herself, a limitation of her approach is the fact that the protocol allows only purely passive eavesdroppers to listen in on the communication. This choice simplifies the formulation of compositional results, as an eavesdropper cannot change the course of communication, *e.g.* by conducting a communication in which it pretends to be an honest participant. Her approach does provide attractive compositional reasoning techniques. We anticipate that we model an active intruder in a TA setting in [7].

Finally, another related approach can be found in [44,45], where Oheimb *et al.* introduce Interactive State Machines (ISMs)—yet another extension of IOA—and apply them to security analysis. In fact, ISMs are used to model and analyze the classic Needham-Schroeder public-key authentication protocol in the version fixed by Lowe [37]. A strong point of this approach is the fact that it allows automatic verification by defining ISMs, and proving theorems, in the theorem prover Isabelle/HOL [42]. What is missing are solid techniques for compositional reasoning over more complex communication protocols. We anticipate that in [7] we define a compositional analysis strategy for TA.

The structure of this paper is as follows. In Section 2 we informally describe how to use TA, after which the above-mentioned papers are surveyed in Sections 3-5. Section 6 concludes the paper.

2 Team Automata

In this section we informally describe the main characteristics of TA and how to use them, rather than defining the framework in all its technical details. For more information on TA the reader is referred to [2,33,50].

To model a system as a TA, first the components have to be identified. Each of them should be given a description in the form of an automaton—an easy to understand state-transition model that moreover forms the basis for system descriptions in a number of model-checking tools [32,34]. Based on the idea of synchronizations of common actions, these components can be connected in order to collaborate. Within each component, a distinction has to be made between internal actions—which are not available for synchronization with other components—and external actions—which can be used to synchronize components and may be subject to synchronization restrictions. By assigning such different roles to actions it is possible to describe many types of collaboration.

Consequently, for each external action separately, a decision is made as to how and when the components should synchronize on this action. If the action is supposed to be a passive action that may not be under the component's local control, then it can be designated as an input action of that component, otherwise as an output action. If such a distinction between the roles of an external action is not necessary, then the choice is arbitrary. A natural option would be to make it an output action in all components in which it occurs. Once the synchronization constraints for each external action have been determined, one may apply, *e.g.*, a maximality principle to construct a unique TA satisfying all constraints. The composition used in IAO, *e.g.*, is mirrored by the so-called *max-ai* TA over a set of CA. In such a max-ai TA the synchronizations between CA that are included are, for all actions, all and only those transitions in which all CA participate that have the action in their alphabet.

The TA framework thus supports component-based system design by making explicit the role of actions and the choice of transitions that govern the collaboration between components. The crucial features are the freedom of choice for the synchronizations collected in the transition relation of a TA and the possibility of synchronizing on output actions. In fact, in order for a TA to be capable of modelling various types of collaboration between its components by synchronizations of common actions, synchronizations between output actions of its components should not be excluded *a priori*, nor should the set of synchronizations be fixed *a priori*. Since these two features allowed the definition of the above-mentioned peer-to-peer and master-slave synchronizations, they were given by [18] as the main reasons for introducing TA to model groupware systems rather than using IOA for that purpose. The last feature distinguishing TA from IOA, finally, is the fact that TA need not be input enabled. Also this feature is motivated from practice. No matter how convenient input enabling may be when modelling reactive systems, it does hinder a realistic modelling of collaborations that involve humans—in fact, Tuttle himself was the first to acknowledge this when he introduced IOA in [49]—while modelling such collaborations was one of the main reasons for the introduction of TA in [18].

3 Access Control Policies

A vital component of the security of any (computer) system or environment is information access control, but this is sometimes done in a rather *ad hoc* or inadequate fashion with no underlying rigorous, formal model. In typical electronic file systems, access rights such as read access and write access are allocated to users on some basis such as ownership or *ad hoc* lists of accessors. Within groupware systems, typically more refined access rights are needed, such as the right to scroll a document that is being synchronously edited by a group in real time. Furthermore, the granularity of access must be more fine grained and flexible in many cases, such as within a software development team. Finally, it is important to control meta access rights. For example, it may be useful for an author to grant another team member the right to grant file access to non-team members (*i.e.* delegation).

In [4], ter Beek *et al.* use a spatial access metaphor based upon work of Bullock *et al.* [9,10]—where access control is governed by rooms, or spaces, in which subjects and objects reside, and the ability of a subject to traverse space in order to get close to an object. A virtual reality is considered in which a user can traverse from room to room by using keyboard keys, the mouse, or fancier devices. It is a natural and simple extension to assume that access control checking happens at the boundaries (doors) between spaces (rooms) when a user attempts to move from one room to another. If the access is OK, then the user can enter and use the resources associated with the newly entered room. By adopting a spatial approach to access control, one exploits a natural part of the environment, making it possible to hide explicit technical security mechanisms from end users through the natural spatial makeup of the environment. These users can then make use of their knowledge of the environment to understand the implicit security policies. Users can thus avoid understanding technical concepts such as access control matrices, which helps to avoid misunderstandings.

In the security literature, authentication deals with verification that the user is truly the person represented, whereas authorization deals with validation that the user has access to the given resource. In [4], only authorization is considered. The goal is to connect the metaphor of spatial access control to the TA framework, and to show how this combination facilitates the identification and unambiguous description of some key issues of access control. To this aim, it is showed how certain spatial access control mechanisms can be made precise and given a formal description using TA. To begin with, information access modelling by granting and revoking access rights is introduced, and it is showed how *immediate* versus *delayed revocation* (does a user immediately lose its access rights when they are revoked—even if it is currently actively using the file—or can a user continue its current activity when its rights have been revoked until it wants to restart this activity—at which moment an authorization check is done to decide whether or not it has the right to restart this activity?) can be formulated.

Subsequently, the study is extended to the more complex issue of meta access control. This means that privileges such as granting and revoking of access rights can themselves be granted and revoked. The notion of meta obviously extends to arbitrary layers. An example of such a multi-layered structure of meta can be seen in the journal refereeing process. The creator of an article may delegate publication responsibilities to co-authors, who may select a journal and grant read access rights to the editor-in-chief, who may grant read access rights to assistant editors, who can then grant and revoke read access to reviewers. However, should revocation of a meta right also revoke the rights that were passed on to others? Here one touches upon the issue of *shallow* versus *deep revocation*. Shallow revocation means that a revoke action does not revoke any of the rights that were previously passed on to others, whereas deep revocation means that a revoke action does revoke all rights previously passed on. Shallow revocation is often the easiest to model, whereas deep revocation is known as a big challenge to model and implement due to the complicated (recursive) situations that may arise [13]. In [4] it is nevertheless demonstrated how TA can be used to unambiguously and concisely model shallow, deep, and even hybrid revocation.

In [3], ter Beek and Bloem undertake an initial attempt to use the model checker Spin [32] for validating some of the TA specifications of [4]. Model checking is an automatic technique to verify whether a system design satisfies its specifications by inspecting its behavior exhaustively, *i.e.* all possible input combinations and states are taken into account. The design is to be given in terms of (finite) automata and the properties that should hold are to be given in terms of logical formulae. The model checking algorithm then checks whether the particular model satisfies the particular property. Spin is one of the best known and most successful model checkers, which was developed at Bell Labs during the last two decades [32]. In [3], the TA specifications of [4] are translated into Promela, which is the input language of Spin. Consequently, the authors verify whether those TA specifications indeed model deep revocation. It turns out that an additional notion of fairness is needed, designating certain states as "illegal" in the sense that a TA is not allowed to remain in those states for more than a limited period of time. Further research in this direction is currently being undertaken.

4 Multicast/Broadcast Communication Protocols

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.* TV broadcasts) or multicast the stream only to designated recipients (*e.g.* pay-per-view TV). Multicast and broadcast data packets are usually sent over unreliable transport protocols, such as the User Data Protocol (UDP). This may cause packet loss, *i.e.* the stream is received incomplete by a part of the recipients.

In [6], the authors of the present paper demonstrate that TA are well suited to model secure multicast/broadcast communication with possible packet loss since they use TA to model an instance of the Efficient Multi-chained Stream Signature (EMSS) multicast protocol family [46]. The usefulness of TA in this context is a consequence of the natural way in which one-to-many (one-to-all) communications typical of multicast (broadcast) sessions can be modelled as synchronizations between the CA constituting a TA. In particular, a multicast composition operator $\|^J$ is defined, which enables the modelling of a multicast protocol involving one sender and n receivers as one-to-J synchronizations between the components of a TA. Recall that such an operator cannot be defined in IOA.

As a final consideration we observe that the basic communication mechanism in most CCS-like process algebras, on the other hand, is pairwise synchronization (in the form of an input/output handshake) between just two processes. This shows why in [41], 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 communication. Moreover, packet loss is modelled by considering a receiver process that nondeterministically chooses whether 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.

5 Towards a TA Framework for the Analysis of Security Properties

In [7], the authors of the present paper define an insecure communication scenario for TA. Their scenario can be used to analyze security properties of cryptographic communication protocols involving two roles, *viz.* an *initiator* TA $\mathcal{T}_S$ and a *responder* TA $\mathcal{T}_R$. Rather than a direct communication between $\mathcal{T}_S$ and $\mathcal{T}_R$, all communication is assumed to flow through an *insecure channel* TA $\mathcal{T}_{IC}$. This insecure channel may release some messages to an *intruder* TA $\mathcal{T}_X$, which in its turn can either listen to or modify (fake) the messages passing through this channel. When verifying security properties for cryptographic communication protocols, it is indeed quite common to include an additional intruder (*à la* Dolev-Yao [15]) 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 with respect to a security property if it satisfies this property despite the presence of the intruder. Abstracting from the cryptographic details concerning the operations according to which messages can be encrypted, decrypted, etc., the insecure scenario is informally described by the TA interactions sketched in Fig. 1.

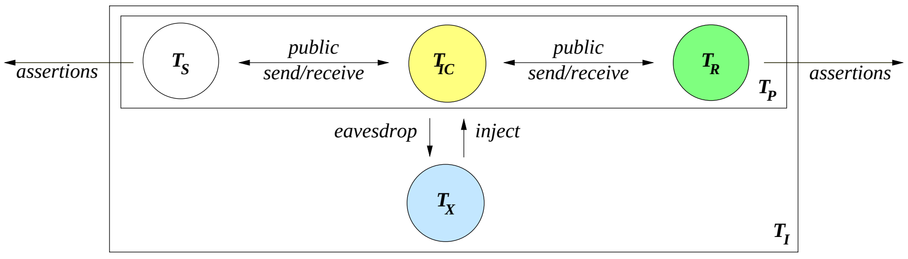

Fig. 1. An insecure communication scenario for TA.

$\mathcal{T}_P$ denotes the TA representing the protocol specification in the absence of the intruder. It is thus defined by hiding all *public send/receive* actions that pass through the insecure channel and then enforcing max-ai synchronization between $\{\mathcal{T}_S, \mathcal{T}_R, \mathcal{T}_{IC}\}$. $\mathcal{T}_P$ is called the max-ai TA over $\{\mathcal{T}_S, \mathcal{T}_R, \mathcal{T}_{IC}\}$ that is obtained after hiding all public send/receive actions. $\mathcal{T}_P$ thus appears as a black box, possibly with some output actions (*assertion*) signalling the successful reception of messages. Usually such signals are used only for verification purposes. $\mathcal{T}_I$ denotes the TA representing the protocol specification in the presence of the intruder. So-called *eavesdrop* and *inject* actions serve as the backdoor for intrusion. This is exactly what is needed to guarantee that the intruder may communicate with $\mathcal{T}_P$ only through the insecure channel. $\mathcal{T}_I$ is thus defined to be the max-ai TA over $\{\mathcal{T}_P, \mathcal{T}_X\}$ that is obtained after hiding all actions the intruder can eavesdrop from and inject back into the insecure channel. In this way, maximal synchronization is enforced also between the intruder and the protocol. This defines an insecure communication scenario for TA by composing a secure communication scenario with an intruder.

In the literature, several efforts have been made to prevent the unauthorized information flow in multilevel computer systems [8], *i.e.* systems where processes and objects are bound to a specific security level. An example from military jargon is the fact that documents are generally hierarchized from unclassified to top secret. The seminal idea of *non interference* proposed in [28] aims at assuring that information can only flow from low levels to higher ones and not vice versa. This notion has been compactly modelled in [43] in terms of automata whose sets of (input) actions are split into low and high classes. This formalization moreover satisfies the condition of Bell and La Padula [8], which forbids the so-called 'read-up' and 'write-down'. In the context of a CCS-like process algebra, on the other hand, the first taxonomy of non-interference-like properties has been uniformly defined in [20,21,22]. In particular, processes in the algebra were divided into high and low processes, according to the level of actions that they can perform. To detect whether an incorrect information flow (*i.e.* from high to low) has occurred, a particular non-interference-like property has been defined, the so-called *Non Deducibility on Compositions* (NDC). NDC essentially says that a process is secure with respect to wrong information flows if its low behaviour in isolation appears to be the same as its low behaviour when interacting with any high-level process. NDC can be reformulated from the world of multilevel systems to the one of network security. See also [24,26], where the low-level process becomes a specification of a cryptographic communication protocol, and the behaviour of the protocol running in isolation is compared with that of the protocol running in parallel with any possible adversary.

As a further step, a *Generalized* NDC (GNDC) has been formulated in [26], in order to encompass in a uniform way many security properties. Informally, a 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 $\alpha(P)$ (with respect to a notion $\triangleleft$ of external observation). This $\alpha(P)$ represents the correct behaviour of P. By varying $\alpha(P)$, several security properties can be defined and analyzed within this generalized schema.[2] In order to embed TA in the well-established analysis framework just described, the authors of the present paper formulated the GNDC schema in terms of TA in [7].

Based on the GNDC schema in terms of TA, a compositional analysis

[2] Recently a slightly extended GNDC schema was defined [25], incorporating the fact that the set of bad behaviours of P may depend on P itself and on the property under scrutiny.

strategy for the above insecure scenario is described. This strategy can be used to verify security properties in the communication protocol modelled by the scenario. In fact, in [7] the GNDC schema in terms of TA, together with the insecure communication scenario for TA, is used to show that integrity is guaranteed in the deterministic (1,2) schema of the EMSS protocol. While this has already been validated in [41], where a CCS-like process algebra was used instead, this particular case study does show the effectiveness of TA for the analysis of security properties. Note that this case study moreover shows that the approaches described in Sections 4 and 5 can be combined in order to provide a compositional analysis strategy for verifying security properties in cryptographic protocols with multicast/broadcast communication. One may comment that [7] tests a theoretical approach over a protocol for which the security property to be verified is already known to be guaranteed. Nevertheless, this case study has been investigated for testing the approach. However, the use of TA is not limited to proving integrity, but also other security properties like secrecy and entity authentication can be verified.

6 Conclusions and Future Work

In this paper we presented a survey of the use of TA for the specification and analysis of some issues from the field of security. More precisely, we have showed how TA can adequately be used to model and verify access control policies, multicast/broadcast communication protocols, and (cryptographic) communication protocols. In many ways, this paper accompanies [33], where a survey of the use of TA for CSCW was presented. An increasing number of papers bears witness to the usefulness of TA in the early design phase of distributed and reactive systems. These examples are not limited to fields like CSCW [5,17,18,36] and security [3,4,6,7,16], but extend to areas like software engineering [19,30,31]. In fact, a spectrum from hardware components to protocols for interacting groups of people has by now been modelled by TA.

The TA framework supports the design of reactive and distributed systems and protocols by making explicit the role of actions and the choice of transitions governing the communication, coordination, cooperation, and collaboration. Moreover, the formal setup and the possibility of a modular design provide analytic tools for the verification of desired properties of complex computer systems. Model-checking techniques, such as those employed in [3], can consequently be used to validate the resulting systems. A goal for the future is to try to automate the currently manual specification and verification of properties in the TA framework. Since TA form an extension of IOA, the *IOA Language and Toolset* [27] may be of help when trying to achieve this goal. This framework provides tool support for defining IOA as well as validating their properties (through theorem proving, model checking, and simulation).

Another goal for the future is to extend the TA framework with time, probability, or both. Such extensions of automata-based formalisms have been well studied in the literature, *e.g.* for IOA [39,48]. In this respect, also the well-developed theory of *timed automata* needs to be mentioned [1,35]. Like their IOA counterparts, timed TA could consider the elapsing of time in the systems they model, whereas probabilistic TA could allow a probabilistic choice of the next state. Both have been extensively used to describe a variety of timing-based algorithms and probabilistic cryptographic protocols, and to prove their properties. Given these existing automata-theoretic results, the extension of TA with time and probability is expected to be a feasible one.

7 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".

Finally, we thank the three anonymous referees for their suggestions that have helped us to improve the presentation of the paper.

References

[1] Alur, R., and D.L. Dill, *A Theory of Timed Automata*, Theoretical Computer Science **126** (1994), 183–235.

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

[3] ter Beek, M.H., and R.P. Bloem, *Model Checking Team Automata for Access Control*, unpublished manuscript, 2003.

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

[5] ter Beek, M.H., 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** (2003), 21–69.

[6] ter Beek, M.H., G. Lenzini, and M. Petrocchi, *Team Automata for Security Analysis of Multicast/Broadcast Communication*, Tech. Rep. 2003-TR-13,

Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, 2003. Presented at WISP'03, 2003 (no formal proceedings).

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

[8] Bell, D.E., and L.J. La Padula, *Secure Computer Systems—Unified Exposition and Multics Interpretation*, Tech. Rep. ESD-TR-75-306, MITRE MTR-2997, 1976.

[9] Bullock, A., and S. Benford, *Access Control in Virtual Environments*, Proc. VRST'97, ACM Press, 1997, 29–35.

[10] Bullock, A., and S. Benford, *An access control framework for multi-user collaborative environments*, Proc. GROUP'99, ACM Press, 1999, 140–149.

[11] Chen, H., and D. Wagner, *MOPS: an Infrastructure for Examining Security Properties of Software*, Proc. CCS'02, ACM Press, 2002, 235–244.

[12] Chen, H., D. Wagner, and D. Dean, *Setuid Demystified*, Proc. Security'02, USENIX, 2002, 171–190.

[13] Dewan, P., and H. Shen, *Flexible Meta Access-Control for Collaborative Applications*, Proc. CSCW'98, ACM Press, 1998, 247–256.

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

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

[16] Egidi, L., and M. Petrocchi, *Modelling a Secure Agent with Team Automata*, Tech. Rep. TR-INF-2004-07-08-UNIPMN, Dipartimento di Informatica, Università degli Studi del Piemonte Orientale *Amedeo Avogadro*, July 2004.

[17] Ellis, C.A., and K.-H. Kim, *A Framework and Taxonomy for Workflow Architectures*, Designing Cooperative Systems: The Use of Theories and Models—Proc. COOP'00, IOS Press, 2000, 99–112.

[18] Ellis, C.A., *Team Automata for Groupware Systems*, Proc. GROUP'97, ACM Press, 1997, 415–424.

[19] Engels, G., and L.P.J. Groenewegen, *Towards Team-Automata-Driven Object-Oriented Collaborative Work*, Formal and Natural Computing, LNCS **2300**, Springer, 2002, 257–276.

[20] Focardi, R., and R. Gorrieri, *A Classification of Security Properties for Process Algebras*, Journal of Computer Security **3, 1** (1994), 5–33.

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

[22] Focardi, R., and R. Gorrieri, *Classification of Security Properties—Part II: Information Flow*, Proc. FOSAD 2001—Tutorial Lectures, LNCS **2171**, Springer, 2001, 331–396.

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

[24] Focardi, R., R. Gorrieri, and F. Martinelli, *Secrecy in Security Protocols as Non Interference*, ENTCS **32**, 2000.

[25] Focardi, R., R. Gorrieri, and F. Martinelli, *Classification of Security Properties—Part II: Network Security*, Proc. FOSAD 2001/2002 Tutorial Lectures, LNCS **2946**, Springer, 2004, 139–185.

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

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

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

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

[30] 't Hoen, P.J., "Towards Distributed Development of Large Object-Oriented Models—Views of Packages as Classes", Ph.D. thesis, Leiden Institute of Advanced Computer Science, Leiden University, 2001.

[31] 't Hoen, P.J., and M.H. ter Beek, *A Conflict-Free Strategy for Team-Based Model Development*, Proc. PDTSD'00, 2000, 720–725.

[32] Holzmann, G.J., "The SPIN Model Checker—Primer and Reference Manual", Addison Wesley, 2003.

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

[34] Kurshan, R.P., "Computer-Aided Verification of Coordinating Processes—The Automata-Theoretic Approach", Princeton Univ. Press, 1994.

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

[36] Lavana, H., "A Universally Configurable Architecture for Taskflow-Oriented Design of a Distributed Collaborative Computing Environment", Ph.D. thesis, Department of Electrical and Computer Engineering, North Carolina State University, 2000.

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

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

[39] Lynch, N.A., *Input/Output Automata: Basic, Timed, Hybrid, Probabilistic, Dynamic,...*, Proc. CONCUR'03, LNCS **2761**, Springer, 1996, 191–192.

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

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

[42] Nipkow, T., L.C. Paulson, and M. Wenzel, "Isabelle/HOL—A Proof Assistant for Higher-Order Logic", LNCS **2283**, Springer, 2002. For an up-to-date description, see `http://isabelle.in.tum.de/`.

[43] Moskowitz, I.S., and O.L. Costich, *A Classical Automata Approach to Noninterference Type Problems*, Proc. CSFW'92, IEEE Press, 1992, 2–8.

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

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

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

[47] Schneider, F.B., *Enforceable Security Policies*, ACM Transactions on Information and System Security **3, 1** (2000), 30–50.

[48] Segala, R., *A Compositional Trace-Based Semantics for Probabilistic Automata*, Proc. CONCUR'95, LNCS **962**, Springer, 1995, 234–248.

[49] Tuttle, M.R., "Hierarchical Correctness Proofs for Distributed Algorithms", Master's thesis, Dept. of Electrical Engineering and Computer Science, MIT, 1987. Tech. Rep. MIT/LCS/TR-387, 1987.

[50] The TA webpage: `http://fmt.isti.cnr.it/ mtbeek/TA.html`. The TA bibliography: `http://liinwww.ira.uka.de/bibliography/Theory/TA.html`.