Dipartimento di Informatica
Università del Piemonte Orientale "A. Avogadro"
Spalto Marengo 33, 15100 Alessandria
http://www.di.unipmn.it

Modelling a Secure Agent with Team Automata

Authors: Lavinia Egidi (lavinia.egidi@mfn.unipmn.it), Marinella Petrocchi (marinella.petrocchi@iit.cnr.it),

TECHNICAL REPORT TR-INF-2004-07-08-UNIPMN
(July 2004)

The University of Piemonte Orientale Department of Computer Science Research Technical Reports are available via WWW at URL `http://www.di.mfn.unipmn.it/`. Plain-text abstracts organized by year are available in the directory

Recent Titles from the TR-INF-UNIPMN Technical Report Series

2004-07 Making CORBA fault-tolerant, Codetta Raiteri D., April 2004.

2004-06 Orthogonal operators for user-defined symbolic periodicities, Egidi, L., Terenziani, P., April 2004.

2004-05 RHENE: A Case Retrieval System for Hemodialysis Cases with Dynamically Monitored Parameters, Montani, S., Portinale, L., Bellazzi, R., Leonardi, G., March 2004.

2004-04 Dynamic Bayesian Networks for Modeling Advanced Fault Tree Features in Dependability Analysis, Montani, S., Portinale, L., Bobbio, A., March 2004.

2004-03 Two space saving tricks for linear time LCP computation, Manzini, G., February 2004.

2004-01 Grid Scheduling and Economic Models, Canonico, M., January 2004.

2003-08 Multi-modal Diagnosis Combining Case-Based and Model Based Reasoning: a Formal and Experimental Analisys, Portinale, L., Torasso, P., Magro, D., December 2003.

2003-07 Fault Tolerance in Grid Environment, Canonico, M., December 2003.

2003-06 Development of a Dynamic Fault Tree Solver based on Coloured Petri Nets and graphically interfaced with DrawNET, Codetta Raiteri, D., October 2003.

2003-05 Interactive Video Streaming Applications over IP Networks: An Adaptive Approach, Furini, M., Roccetti, M., July 2003.

2003-04 Audio-Text Synchronization inside mp3 file: A new approach and its implementation, Furini, M., Alboresi, L., July 2003.

2003-03 A simple and fast DNA compressor, Manzini, G., Rastero, M., April 2003.

2003-02 Engineering a Lightweight Suffix Array Construction Algorithm, Manzini, G., Ferragina, P., February 2003.

2003-01 Ad Hoc Networks: A Protocol for Supporting QoS Applications, Donatiello, L., Furini, M., January 2003.

2002-06 Stochastic modeling, analysis techniques and tools for dependable reactive systems, Codetta Raiteri, D., Bobbio, A., October 2002.

2002-05 Stochastic modeling, analysis techniques and tool for dependable reactive systems, Bernardi, S., Gribaudo, M., Bobbio, A., October 2002.

2002-04 Interactive MPEG video streaming over IP-Networks: a performance report, Furini, M., Roccetti, M., September 2002.

Modelling a Secure Agent with Team Automata

Lavinia Egidi
Dipartimento di Informatica
Università degli Studi del Piemonte Orientale
Spalto Marengo 33, Alessandria, Italy
lavinia@mfn.unipmn.it

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

July 9, 2004

Abstract

We use Team Automata in order to model a protocol [9] for securing agents in a hostile environment. Our study focuses on privacy properties of the agents. We use the framework to prove a result from [9]. As a by-product, our analysis gives some initial insight on the limits of the protocol. From a different perspective, this study continues a line of research on the expressive power and modelling capabilities of Team Automata. To the best of our knowledge, this is the first attempt to use Team Automata for the analysis of privacy properties.

1 Introduction

Agent technology is assuming a central role in various areas of computer science. Mobile agents are indeed a powerful tool for limiting data traffic or managing remote service provision. But since mobile agents are software meant to run on foreign hosts, various security issues arise in their respect. On the one side, hosts must be protected from non trusted agents that might carry malicious code. This is the easy side of the question, and is addressed with computer security techniques. In contrast, it is much harder to protect a mobile agent from a hostile environment. While an agent executes on a host, its code must be in the clear; if it needs to use sensitive data that it carries along, this data must be in the clear as well. If it is stored in an encrypted form, it must be decrypted prior to use, and therefore the appropriate decryption key must be available to

the agent. This suggests that usual cryptographic tools cannot protect an agent from being robbed or spied upon.

But, although some vulnerabilities cannot be eliminated (the agent can be killed or bogus data can be supplied to it), it would be very appealing to provide data privacy and integrity mechanisms to mobile agents. Enhancing them with security features can result in a very powerful and effective way of handling services on potentially hostile resources.

Because of the wide range of applications that can be imagined for a secure agent, the issue is currently a hot topic in research. Recent approaches look for a solution by carrying the idea of encryption to an unusual level; briefly stated, although the agent code itself remains in the clear, the function computed by the agent is transformed so that the agent's behaviour is incomprehensible to an observer that doesn't have a key for interpretation. The first proposal in this direction is due to Sander and Tschudin [14]. Their technique has been generalized by Cachin et al. [9] who use the idea of garbled circuits by Yao [15]. It must be noted that these approaches are still pioneer solutions to the security problem posed above, in that the security goals achieved are still very restrictive and the agent model to which they apply is somewhat awkward.

As a first step towards a comprehension of the potentialities of these methods, and of intrinsic limitations of software agents from the point of view of security, we formalize the protocol of Cachin et al. [9] using Team Automata (TA).

TA are inspired by—and form an extension of—Input/Output automata (IOA) [12]. TA form a flexible framework for modelling communication between system components. 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. In particular, given a set of CA, there is no such thing as the unique TA composed over that set of CA. Rather, a whole range of TA, distinguishable only by their synchronizations, can be composed over this set of CA.

The rigorous setup of the TA framework 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.

In particular, TA have been proved to cover the specification of several access control strategies of [3], as well as the ongoing work on developing a TA framework for the analysis of security properties—which was initiated in [5] and further developed in [6]. In [7], it is presented a survey of the use of TA in the field of security. We continue this line of research, by exploiting their flexibility and intuitive modelling of a multi-host environment.

The protocol of Cachin et al. [9] is based on the idea of entrusting data to an agent in the form of a circuit that evaluates to a single output. The circuit is obtained as a cascade of components each one constructed by one of the hosts visited by the agent. In [9], it is proven that the protocol preserves the privacy of all actors' inputs.

We contribute a TA based analysis of this privacy property. To the best of our knowledge, TA have not been used before in the study of privacy.

Our analysis also has the merit of providing a high level model of the actors' behaviour and interaction, abstracting out from cryptographic details, giving a clearer insight of the protocol. Our model, interestingly, naturally represents the agent as a set of actions as opposed to an entity *per se*. This suggests that the protocol does not respect the object oriented spirit of agents. On the other hand, taking the perspective of the agent's source, it models the source's view of the system according to the intuition that the source delegates tasks fully to the agent.

In the following section we give a high level description of the protocol and recall the relevant results about garbled circuits and the protocol itself. In Section 3 we overview the basic facts about TA. Section 4 is devoted to the model and to the formal analysis of privacy features. We conclude with some remarks.

2 The protocol

The soundness and privacy properties of the protocol of [9] follow from mathematical results based on standard cryptographic assumptions. We abstract out from the details and are interested in the behaviour of the actors (the agent, the agent's source and the hosts that are visited by the agent) at a higher level. Our description of the protocol focuses on the interactions between the actors. We give as much insight on the cryptographic bases on which the protocol relies, as is sufficient for our arguments. We refer to [9] for a detailed description.

We confine ourselves to the simpler case of the "honest but curious model," in which actors are supposed to follow the protocol correctly, but they might try to learn the private inputs of the other parties.

The goal of the design is a secure agent that travels through many hosts collecting sensitive information and then back to its source; back home, it will be able to deliver the result of a computation on inputs collected at the various hosts together with the source's input. The security feature that the protocol aims at is privacy of all the inputs, that is no party learns the inputs of any other party.

The idea is to combine in a cascade Yao style garbled circuits [15]. The software agent travels from host to host collecting private information in the form of a (portion of) garbled circuit. The circuit (potentially software) is actually data, since it can only be evaluated to a single value once it has been brought back to the agent's source.

2.1 Garbled circuits

A garbled circuit is a generalization of a circuit, with the following properties:

- each wire can carry one of two specified random strings (not just bits 0/1), the random strings changing from wire to wire (the pair of strings on each wire have semantical interpretation 0 and 1);
- for each gate a specific computational rule is given, that defines how the random strings in input are to be combined to produce the output, which is again a random string (the semantical interpretation of a gate is a NAND or a XOR, for instance).

The *garbled inputs* are the random strings of each input wire whose semantical value is the value of the corresponding input bit. The *decoding* of the output is a translation of the random strings on the output wires to their semantical meanings. The *garbled circuit* is a description of the structure of the circuit together with computational rules for each node but *no information on the random strings carried by each wire.*

The following holds (see [13]):

Lemma 1 (Indistinguishability) *For any two actors C and D knowing the garbled circuit, if C only knows the garbled version of D's input (and* not *the other random strings carried by D's input wires) garbled circuit evaluation will not disclose more information than if C ran the protocol assuming any random input for D.*

2.2 The Wannabe Traveller

For a lighter exposition, and without loss of generality, we present the protocol in the specific setting of an actor W (the Wannabe Traveller) who dispatches an agent in quest of the best offer for a holiday on a tropical island.

The agent visits travel agencies Ag_j, chosen according to some policy that we do not specify here. At each agency, it browses the catalogue and requests the best offer for a holiday matching conditions on the destination, the period, the services, etc., that W requires.

We assume that travel agencies want the privilege of tailoring their offers to the specific client, and therefore prefer that the offer be known only if it is highly likely to be accepted. Moreover each agency does not want that its offer be known to competitors. On the other hand, W does not want to disclose in advance her budget, to avoid that travel agents use it as an information to adjust their offer. Assuming correctness, this leads us to the following definition of the privacy goals we aim at:

Definition 1 (Privacy) *W's agent respects privacy if*

1. *W cannot determine any other offer but the lowest one less than or equal to her budget, if it exists;*
2. *each agency cannot learn W's budget, nor the offer of any other agency.*

2.3 The Wannabe-Traveller Protocol

Our analysis is focused on the privacy aspects. Therefore we only consider the agent's functionality related to privately (in the sense described above) conveying to W the best offer. Also, we will not model the agent itself as a separate entity. Rather, the agent is represented by a sequence of actions which it repeats identically at each host visited: the collection of sensitive data.

For simplicity, we first consider the case of an agent that visits a single host. W is the source of the agent, and Ag is the host that the agent visits.

In our setting, Ag constructs a garbled circuit that on input $\langle id_1, x\rangle, \langle id_2, y\rangle$, outputs computes $\langle id_{min}, \min(x,y)\rangle$, where $id_{min} = id_1$ if $x = \min(x,y)$ and is id_2 otherwise. We call this function *tagged minimum*. (The description and analysis that follow are absolutely independent of the specific function computed by the garbled circuit.)

Let $input_W$ (resp. $input_{Ag}$) be W's (resp. Ag's) private input.

In order to evaluate the circuit on $\langle id_W, input_W\rangle$ and $\langle id_{Ag}, input_{Ag}\rangle$, W must learn the garbled circuit, the decoding information of the output and the values of her and Ag's garbled inputs. Ag must not learn W's input. In order to transfer to W the garbled inputs corresponding to W's input $\langle id_W, input_W\rangle$, without Ag's learning the value of the input itself, the two parties use an oblivious transfer (OT) protocol [8]. (Also see [9] for the implementation of a one-round oblivious transfer; we assume that W and Ag share a pseudorandom generator and a seed.)

Let β be W's committal data for OT, referred to $\langle id_W, input_W\rangle$. Let $GC = gc(\langle id_{Ag}, input_{Ag}\rangle, OT(\beta))$ be the garbled circuit computed by Ag, with Ag's input hardwired into it, and information $OT(\beta)$ attached to it, for obliviously transferring to W her garbled input. We denote by *decode* the decoding information for the output.

Then the protocol is as follows:

$$\begin{array}{ll} W \longrightarrow Ag: & \beta \\ Ag \longrightarrow W: & \langle GC, decode\rangle \end{array}$$

W computes the single value $tagged\_min(\langle id_{Ag}, input_{Ag}\rangle, \langle id_W, input_W\rangle)$.

Assuming correctness, the protocol guarantees privacy in the sense of Definition 1:

Lemma 2 (Privacy–two parties) *In the honest but curious model, assuming correctness of the protocol, the two party protocol above guarantees privacy in the sense of Definition 1.*

See [9] for a proof.

In the general case, the agent visits many hosts. We assume, without loss of generality, that the agent travels from W to Ag_1 to Ag_2 and so on, and then back from Ag_n to W.

We generalize and complete the notation that we used for the two-party case. Let $input_W$ be the private input of W, and $input_j$ be the private input of Ag_j. As above, let β be W's commitment to $\langle id_W, input_W\rangle$.

Ag_1 computes a garbled circuit $GC_1 = gc(\langle id_1, input_1\rangle, OT(\beta))$ as in the two-party protocol above. It then forwards $\langle GC_1, decode_1\rangle$ to Ag_2. (The OT data for Alice attached to GC_1 will be forwarded to Alice along with the garbled circuit.)

All other agencies will compute garbled circuits for the tagged minimum function and combine it in cascade one after the other.

For $j > 1$, let $gc_j = gc(\langle id_j, input_j\rangle, transl(decode_{j-1}))$ be the garbled circuit computed by Ag_j, with input $\langle id_j, input_j\rangle$ hardwired to it, and translation information $transl(decode_{j-1})$ for translating the garbled output of the cascade of circuits GC_{j-1} computed by agencies Ag_1 through Ag_{j-1} to a garbled input for gc_j. Then, GC_j is gc_j concatenated to GC_{j-1}. Let $decode_j$ be the instructions for decoding its output.

Ag_j forwards to Ag_{j+1} $\langle GC_j, decode_j\rangle$.

W receives the output of AG_n as if she were an AG_{n+1}, and evaluates it.

The protocol is summarized below ($W \longrightarrow *$ describes W's output of the final value).

$W \longrightarrow Ag_1$:	β	
$Ag_j \longrightarrow Ag_{j+1}$:	$\langle GC_j, decode_j\rangle$	$j = 1,\ldots,n-1$
$Ag_n \longrightarrow W$:	$\langle GC_n, decode_n\rangle$	
$W \longrightarrow *$:	$eval(GC_n, decode_n, \beta)$	

Lemma 3 (Cascade of garbled circuits) *For all $j = 1, \ldots, n$,*

1. *a polynomially bounded actor cannot infer the private inputs of W and $Ag_1, \ldots, Ag_j$ from the garbled circuit $\langle GC_j, decode_j\rangle$;*
2. *moreover, with knowledge of W's (garbled) input, a polynomially bounded actor cannot infer the private inputs of $Ag_1, \ldots, Ag_j$.*

Sketch For $j = 1$, the thesis follows from Lemma 2. For $j > 1$, it can be proven inductively, based on Lemma 1.

We use TA to prove the following privacy property:

Theorem 1 (Privacy—many parties) *In the honest but curious model, assuming correctness of the protocol, the multi-party protocol above guarantees privacy in the sense of Definition 1.*

The proof of the theorem is given in the sequel as it follows from our TA analysis.

3 Team Automata

In this section, we describe the main characteristic of TA. In particular, we introduce some technical details that will be useful throughout the paper. For more information on TA the reader is referred to [1, 2, 10]. Further, we assume some familiarity with automata theory.

A TA $\mathcal{T}$ consists of component automata (CA)—ordinary automata without final states and with a distinction of their sets of actions into input, output, and internal actions—combined in a coordinated way such that they can perform shared actions. Internal actions have strictly local visibility and cannot be used for communication with other CA, while input and output actions together form the external actions that are observable by other CA and that are used for communication between CA. Thus, when composing TA over a set of CA, the internal actions of the CA in the set must be private. In particular, each action that is output (resp. internal) for one or more of the CA constituting a TA becomes an output (resp. internal) action of the TA. The input actions of the CA that do not occur at all as output actions of any of the CA, become the input actions of the TA. Hereafter, we let $\Sigma^{\mathcal{T}}_{inp}, \Sigma^{\mathcal{T}}_{out}, \Sigma^{\mathcal{T}}_{int}$ denote the pairwise disjoint sets of input, output and internal actions of $\mathcal{T}$. Moreover, $\Sigma^{\mathcal{T}}_{ext} = \Sigma^{\mathcal{T}}_{inp} \cup \Sigma^{\mathcal{T}}_{out}$ denotes the set of external actions of $\mathcal{T}$. Finally, $\Sigma^{\mathcal{T}}_{com}$ denotes the set of output actions of $\mathcal{T}$ involved in actual communications between CA in a TA.

During each clock tick, the CA within a TA can simultaneously participate in one instantaneous action, *i.e.,* synchronize on this action, or remain idle. CA can thus be combined in a loose or more tight fashion depending on which actions are to be synchronized, and when. For each external action separately, a decision is made as to how and when the components should synchronize on this action. Each choice of synchronizations thus defines a TA. Every TA is again a CA, which, in turn, can be used in an iteratively composed TA.

Sometimes it can be useful 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. To this aim, we introduce here the *hide* operator: $\mathrm{hide}_\Gamma(\mathcal{T})$ is the TA in which the subset Γ of external actions of the TA $\mathcal{T}$ have become unobservable for other TA, having been turned into internal actions.

It may sometimes be useful to construct unique TA of a specified type. In [4] several fixed strategies for choosing the synchronizations of a TA were defined, each leading to a uniquely defined TA. From those, we use here a maximality principle that will be used in the rest of the paper. Informally, the so called *max-ai* TA is the TA in which the synchronization is defined on all and only those transitions in which, for each action, all CA featuring that action participate to the transition. The *max-ai* TA over a set $\{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$ of CA is denoted as $|||\{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$.

Let Σ and Γ be two sets of symbols. Then, the morphism $\mathrm{pres}_{\Sigma,\Gamma} : \Sigma \to \Gamma^*$, defined by $\mathrm{pres}_{\Sigma,\Gamma}(a) = a$ if $a \in \Gamma$ and $\mathrm{pres}_{\Sigma,\Gamma}(a) = \lambda$ otherwise, preserves the symbols from Γ and erases all other symbols. We discard Σ when no confusion can arise.

Let $\mathcal{T}$ be a TA over a set of CA. Then, the Γ-behaviour of $\mathcal{T}$, denoted as $\mathbf{B}^\Gamma_{\mathcal{T}}$, is defined as usual in automata theory, $\mathbf{B}^\Gamma_{\mathcal{T}} = \mathrm{pres}_\Gamma(\mathbf{C}_{\mathcal{T}})$, with set $\mathbf{C}_{\mathcal{T}}$ of computations of $\mathcal{T}$ consisting of all the sequences $\alpha = q_0 a_1 q_1 \ldots a_n q_n$, where $n \geq 0$ and q_0 is an initial state, q_i, are states, a_i are actions and (q_{i-1}, a_i, q_i) are transitions.

Along with this general notion of behaviour, other notions can be defined. When $\Gamma = \Sigma_{out}^{\mathcal{T}}$, then $\mathbf{B}_{\mathcal{T}}^{\Sigma_{out}}$ is the output behaviour of $\mathcal{T}$. By opportunely choosing Γ, also the input and the internal behaviour of $\mathcal{T}$ can be defined.

4 The Wannabe-Traveller Protocol Modelled by TA

We now show how TA can be used to model the Wannabe Traveller protocol. We model the Wannabe Traveller W by a CA $\mathcal{T}_W$, the set $\{Ag_j \mid 1 \leq j \leq n\}$ of travel agencies by CA $\mathcal{T}_{Ag_1}, \ldots, \mathcal{T}_{Ag_n}$.

Let Input denote the set of pairs $\{\langle id_j, input_j\rangle \mid 1 \leq j \leq n\} \cup \{\langle id_0, input_W\rangle\}$ where $input_j$ (resp. $input_W$) is a string that is private to Ag_j (resp. to W).

Let Computed denote the set of garbled circuits.

Let β be W's OT commitment data. Let Decode denote the set $\{decode_j \mid 0 \leq j \leq n\} \cup \{\beta\}$ where $decode_j$ $(j = 1, \ldots, n)$ is the decoding information for the output of circuit GC_j.

Then $\mathcal{T}_{Ag_j}$ uses the function gc : Input $\times$ Decode $\rightarrow$ Computed to compute the garbled circuit gc_j and the function $||$: Computed $\times$ Computed $\rightarrow$ Computed to build up the circuit GC_j consisting of the cascade of garbled circuits gc_1 through gc_j.

Let Result = Input. Then, $\mathcal{T}_W$ evaluates the final result using the function $eval$: Computed $\times$ Decode $\times$ Input $\rightarrow$ Result.

For each $j = 1, \ldots, n$, define $P_j = \langle GC_j, decode_j\rangle$ and $P_0 = \beta$. Then, Messages denotes the set $\{P_j \mid 0 \leq j \leq n\} \cup$ Result.

We specify TA in the way IOA are commonly defined [11, 12]. 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.

In all the specifications, we explicitly prohibit loops, *i.e.,* we allow each action to be performed only once. See, for example, the specification of $\mathcal{T}_{Ag_1}$. As soon as $\mathcal{T}_{Ag_1}$ has received P_0, then precondition $P_0 \notin$ received prevents this action to be executed again.

$\mathcal{T}_{Ag_j}$ — Travel Agencies, $j = 1, \ldots, n$

Actions

Inp: $\{P_{j-1}\}$

Out: $\{P_j\}$

Int: $\{Compute_j\}$

States

received, sent $\subseteq$ Messages, computed $\subseteq$ Computed, all initially $\varnothing$

Transitions

P_{j-1}
Pre: $P_{j-1} \notin$ received
Eff: received := received $\cup \{P_{j-1}\}$

$Compute_j$
Pre: $P_{j-1} \in$ received $\wedge\ GC_j \notin$ computed
Eff: computed := computed $\cup \{GC_j\}$

P_j
Pre: $GC_j \in$ computed $\wedge\ P_j \notin$ sent
Eff: sent := sent $\cup \{P_j\}$

The input behaviour $\mathbf{B}^{\Sigma_{inp}}_{\mathcal{T}_{Ag_j}}$ of $\mathcal{T}_{Ag_j}$ $(j = 1, \ldots, n)$ consists of P_{j-1}. When $\mathcal{T}_{Ag_j}$ receives message P_{j-1}, then $\mathcal{T}_{Ag_j}$ is able to perform an internal computation leading to an internal behaviour $\mathbf{B}^{\Sigma_{int}}_{\mathcal{T}_{Ag_j}}$ consisting of $Compute_j$. Finally, the output behaviour $\mathbf{B}^{\Sigma_{out}}_{\mathcal{T}_{Ag_j}}$ of $\mathcal{T}_{Ag_j}$, $(j = 1, \ldots, n)$ consists of P_j.

We continue with the specification of $\mathcal{T}_W$. It is capable to output a commitment β to $input_W$. Then, it is capable of receiving as input behaviour the last circuit and the last decoding instructions to evaluate the final result min by starting from what she has received and from $input_W$, by means of function $eval$. Finally, $\mathcal{T}_W$ outputs the final result min.

$\mathcal{T}_W$ — Wannabe Traveller

Actions
Inp: $\{P_n\}$
Out: $\{P_0\} \cup \{min\}$
Int: $\{Eval_{GC_n}\}$

States
received, sent $\subseteq$ Messages, result $\subseteq$ Result, all initially $\varnothing$

Transitions
P_n
Pre: $P_n \notin$ received
Eff: received := received $\cup \{P_n\}$

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

$Eval_{GC_n}$
Pre: $P_n \in$ received $\wedge\ min \notin$ result
Eff: result := result $\cup \{min\}$

min

Pre: $min \in \mathsf{result} \wedge min \notin \mathsf{sent}$
Eff: $\mathsf{sent} := \mathsf{sent} \cup \{min\}$

The input behaviour $\mathbf{B}_{\mathcal{T}_W}^{\Sigma_{inp}}$ of $\mathcal{T}_W$ is clearly represented by P_n. When $\mathcal{T}_W$ receives message P_n, then $\mathcal{T}_W$ is able to perform an internal computation leading to an internal behaviour $\mathbf{B}_{\mathcal{T}_W}^{\Sigma_{int}} = Eval_{GC_n}$. Finally, the output behaviour $\mathbf{B}_{\mathcal{T}_W}^{\Sigma_{out}}$ is $\{P_0 min, min P_0\}$.

Now, we enforce maximal synchronization between the traveller and the agencies. Thus, the max-ai TA over $\{\mathcal{T}_W, \mathcal{T}_{Ag_j} \mid 1 \leq j \leq n\}$, denoted by $\mathcal{T}_{WT}$, is defined as

$$\mathcal{T}_{WT} = |||\{\mathcal{T}_W, \mathcal{T}_{Ag_j} \mid 1 \leq j \leq n\},$$

which formalizes the Wannabe Traveller protocol. From the way CA are composed, the resulting team has no input actions, while it has the union of the output (internal) actions of $\mathcal{T}_W$ and the $\mathcal{T}_{Ag_j}$'s as its output (internal) actions.

4.1 Privacy

In this section, we show, through the use of TA, that W's agent respects privacy, in the sense of Definition 1, in the multiparty case ($n > 1$).

We abstract from the syntax details concerning the operations according to which messages can be manipulated, but we assume the presence of an inference system (defined by a derivation operator $\vdash$) that implements these operations. By applying operations from this system 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.

We restrict the initial knowledge of an automaton A to be bound to a specified set of messages ϕ_A. This informally means that the automaton should be able to produce, by means of only its internal functioning, at most the messages contained in $\mathcal{D}(\phi_A)$. More specifically, when considered as a stand-alone component, the automaton can only execute output actions belonging to $\mathcal{D}(\phi_A)$.

The initial knowledge can be increased to the set ϕ'_A during the execution of the protocol by the messages the automaton receives. Accordingly, the automaton knowledge becomes at most $\mathcal{D}(\phi'_A)$.

We use this notion of knowledge to model privacy. (In order to restrict in a realistic way the inference power of an automaton, we assume, as usual, polynomial boundedness.)

Throughout the analysis, we abstract from the internal computations of the single automata. This is justified by the following: since we are interested in privacy properties, we care about the information flow between the principals, rather than about their internal computations. Thus, in the following we restrict our survey to analyze external actions of our system.

First, we show that W cannot determine any other offer but the lowest one less than or equal to her budget, if it exists (Definition 1(i)).

We must analyze how the knowledge of W is altered in the course of protocol execution. We want to highlight the interactions of W with the rest of the system. Therefore, since we choose to take W's standpoint, communications between agencies, and any distinction among them, are of no interest. So, we combine agencies into a unique block that interacts with W in a way that is indistinguishable from the original system.

We obtain this by defining $\mathcal{T}_{\overline{W}}$ as the max-ai TA over $\{\mathcal{T}_{Ag_j} \mid 1 \leq j \leq n\}$ that is obtained after hiding the actions

$$\Sigma_{com} = \bigcup_{j=1}^{n-1} \Sigma_{com}^{j},$$

i.e., all messages that the travel agencies exchange with each other:

$$\mathcal{T}_{\overline{W}} = \mathrm{hide}_{\Sigma_{com}}(|||\{\mathcal{T}_{Ag_1}, \ldots, \mathcal{T}_{Ag_n}\})$$

Thus, $\mathcal{T}_{\overline{W}}$ appears as a black box, with some input and output actions it will use to interact with the environment. In our setting W plays the role of the environment. Intuitively, this reflects the nature of the protocol itself: W delegates to its agent the choice of the agencies to visit, and does not need to (and cannot) know details of the agent's transactions with them.

Proposition 1

$$\mathbf{B}_{\mathcal{T}_{\overline{W}}}^{\Sigma_{out}} = \mathbf{B}_{\mathcal{T}_{Ag_n}}^{\Sigma_{out}}.$$

Proof The equality follows from the construction of $\mathcal{T}_{\overline{W}}$.

We can now use Proposition 1 to prove the part of Theorem 1 relative to Definition 1(i):

Proof of Theorem 1 (part 1):

By construction, the initial knowledge of $\mathcal{T}_W$ is bound to $\phi_W = \{\beta, input_W\}$. By definition of the automaton knowledge, the only way in which $\mathcal{T}_W$ can significantly increase its knowledge is by performing input actions. To correctly model the protocol we must impose:

$$\mathbf{B}_{\mathcal{T}_W}^{\Sigma_{inp}} = \mathbf{B}_{\mathcal{T}_{Ag_n}}^{\Sigma_{out}},$$

and therefore, by Proposition 1

$$\mathbf{B}_{\mathcal{T}_W}^{\Sigma_{inp}} = \mathbf{B}_{\mathcal{T}_{\overline{W}}}^{\Sigma_{out}}. \tag{1}$$

From the way $\mathcal{T}_{Ag_n}$ is composed, it follows that $\Sigma_{out}^{\mathcal{T}_{Ag_n}} = \{\langle GC_n, decode_n\rangle\}$. From Section 4, it follows that $\langle GC_n, decode_n\rangle$ will be executed, and it will be executed only once. Thus, $\mathbf{B}_{\mathcal{T}_{Ag_n}}^{\Sigma_{out}} = \langle GC_n, decode_n\rangle$.

The latter, Proposition 1 and Equation (1) imply that the knowledge of $\mathcal{T}_W$ becomes at most $\mathcal{D}(\phi'_W)$, with $\phi'_W = \{\beta, input_W, \langle GC_n, decode_n\rangle\}$. Then,

by Lemma 3, and since we are assuming correctness (*i.e.*, GC_n is exactly the garbled circuit computing the (tagged) minimum among all of the private inputs), we conclude that, if W is polynomially bounded, the privacy property of Definition 1(i) holds. □

The proof of the second privacy property (Definition 1(ii)) is very similar.

This time we take the standpoint of Ag_j, for any fixed $j \in \{1, \ldots, n\}$. Again we view the rest of the system as a unique block that interacts with Ag_j in a way that is undistinguishable from the way that the collection of the single actors interact with it. To this end we build the following TA:

$$\mathcal{T}_{\overline{Ag_j}} = \mathrm{hide}_{\mathcal{C}}(\,|||\,\{\mathcal{T}_W, \mathcal{T}_{Ag_1}, \ldots, \mathcal{T}_{Ag_{j-1}}, \mathcal{T}_{Ag_{j+1}}, \ldots, \mathcal{T}_{Ag_n}\})$$

where

$$\mathcal{C} = \bigcup_{i=1}^{j-2} \Sigma^i_{com} \cup \bigcup_{i=j+1}^{n} \Sigma^i_{com} \cup \Sigma^W_{com} \cup \Sigma^W_{out}.$$

We prove for $\mathcal{T}_{\overline{Ag_j}}$ a result analogous to Proposition 1:

Proposition 2

$$\mathbf{B}^{\Sigma_{out}}_{\mathcal{T}_{\overline{Ag_j}}} = \mathbf{B}^{\Sigma_{out}}_{\mathcal{T}_{Ag_{j-1}}}, \ \textit{if } j > 1,$$

$$\mathbf{B}^{\Sigma_{out}}_{\mathcal{T}_{\overline{Ag_j}}} = \mathbf{B}^{\Sigma_{out}}_{\mathcal{T}_W}, \ \textit{if } j = 1.$$

Proof The equalities follow from the construction of $\mathcal{T}_{\overline{Ag_j}}$.

We can now complete the proof of Theorem 1:

Proof of Theorem 1 (part 2):

By construction, the initial knowledge of $\mathcal{T}_{Ag_j}$ is bound to $\phi_{Ag_j} = \{input_j\}$. By definition of the automaton knowledge, the only way in which $\mathcal{T}_{Ag_j}$ can significantly increase its knowledge is by performing input actions. To correctly model the protocol we must impose:

$$\mathbf{B}^{\Sigma_{inp}}_{\mathcal{T}_{Ag_j}} = \mathbf{B}^{\Sigma_{out}}_{\mathcal{T}_{Ag_{j-1}}}, \text{ if } j > 1,$$

and

$$\mathbf{B}^{\Sigma_{inp}}_{\mathcal{T}_{Ag_j}} = \mathbf{B}^{\Sigma_{out}}_{\mathcal{T}_W}, \text{ if } j = 1.$$

Therefore, by Proposition 2

$$\mathbf{B}^{\Sigma_{inp}}_{\mathcal{T}_{Ag_j}} = \mathbf{B}^{\Sigma_{out}}_{\mathcal{T}_{\overline{Ag_j}}}. \tag{2}$$

From the way $\mathcal{T}_{Ag_{j-1}}$ is composed, it follows that $\Sigma^{\mathcal{T}_{Ag_{j-1}}}_{out} = \{\langle GC_{j-1}, decode_{j-1}\rangle\}$. From Section 4, it follows that $\langle GC_{j-1}, decode_{j-1}\rangle$ will be executed, and it will be executed only once. Thus, $\mathbf{B}^{\Sigma_{out}}_{\mathcal{T}_{Ag_{j-1}}} = \langle GC_{j-1}, decode_{j-1}\rangle$. From similar arguments, it follows that $\mathbf{B}^{\Sigma_{out}}_{\mathcal{T}_W} = \beta$.

The latter, Proposition 2 and Equation (2) imply that the knowledge of $\mathcal{T}_{Ag_j}$ becomes at most $\mathcal{D}(\phi'_{Ag_j})$, with

$$\phi'_{Ag_j} = \begin{cases} \{input_{Ag_j}, \langle GC_{j-1}, decode_{j-1}\rangle\} & \text{if } j > 1 \\ \{input_{Ag_j}, \beta\} & \text{if } j = 1. \end{cases}$$

Then, by Lemma 3 (if $j > 1$) or Lemma 2 (if $j = 1$) and since we are assuming correctness, we conclude that, if all actors are polynomially bounded, the privacy property of Definition 1(ii) holds.

This concludes the proof of Theorem 1. □

5 Conclusions and Future Work

We propose a way of modelling the secure agents of [9], in the framework of Team Automata. We investigate a possible way of analyzing privacy properties with Team Automata.

This research is a preliminary step towards the understanding and possibly generalization of techniques for securing mobile autonomous agents. In our analysis we targeted a specific privacy property, which is the core of the proposal of [9], but we aimed at a broader study of the potentials of such an approach. The insight we gain clearly underlines a weakness of the current results in the area. Indeed, it emerges in a natural way, that the protocol we study is not agent oriented in spirit, but it rather offers a means of adding a security layer over agent technologies. Our impression is that such an approach cannot carry very far. In related ongoing research, we are exposing the computational cost of this methodology.

References

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

[2] M.H. ter Beek. 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] 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.

[4] 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.

[5] M.H. ter Beek, G. Lenzini, and M. Petrocchi. Team Automata for Security Analysis of Multicast/Broadcast Communication. Technical Report 2003-TR-13, ISTI-CNR, 2003. Presented at WISP'03.

[6] M.H. ter Beek, G. Lenzini, and M. Petrocchi. A framework for security analysis with team automata. Technical Report TR-CTIT-04-13, University of Twente, 2004. Presented at the DIMACS Workshop on Security Analysis of Protocols 2004.

[7] M.H. ter Beek, G. Lenzini, and M. Petrocchi. Team automata for security –a survey–. Technical Report IIT TR-06/2004, Istituto di Informatica e Telematica - CNR, 2004.

[8] M. Bellare and S. Micali. Non-interactive oblivious transfer and applications. In *Proc. CRYPTO'89*, LNCS 435, pages 547–557. Springer-Verlag, 1989.

[9] C. Cachin, J. Camenisch, J. Kilian, and J. Müller. One-round secure computation and secure autonomous mobile agents. In *Proc. ICALP'00*, LNCS 1853, pages 512–523. Springer-Verlag, 2000.

[10] 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. Springer-Verlag, 2003.

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

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

[13] P. Rogaway. *The round complexity of secure protocols.* PhD thesis, MIT, Cambridge, Massachusetts, 1991.

[14] T. Sander and C.F. Tschudin. Protecting mobile agents against malicious hosts. In *Proc. Mobile Agents and Security*, LNCS 1419, pages 44–60. Springer-Verlag, 1998.

[15] A.C. Yao. How to generate and exchange secrets. In *Proc. 27th FOCS*, pages 162–167. IEEE, 1986.

Notation

For convenience of the reader, we summarize the notation that we used to describe the protocol.

- $tagged\_min(\langle t_1, x_1\rangle, \langle t_2, x_2\rangle) = \langle t_{min}, \min(x_1, x_2)\rangle$, where

$$t_{min} = \begin{cases} t_1 & \text{if } x_1 = \min(x_1, x_2) \\ t_2 & \text{otherwise;} \end{cases}$$

- $input_W$ is the private input of W;
- $input_{Ag}$ is the private input of Ag in the two-party protocol;

- $input_j$ is the private input of Ag_j in the multi-party protocol;
- β is W's committal data for OT, referred to $\langle id_W, input_W \rangle$.
- $GC = gc(\langle id_{Ag}, input_{Ag} \rangle, OT(\beta))$ is the garbled circuit computed by Ag, with Ag's input hardwired into it, and information $OT(\beta)$ attached to it, for obliviously transferring to W her garbled input, in the two-party case;
- *decode* is the decoding information for the output of GC;
- $GC_1 = gc(\langle id_1, input_1 \rangle, OT(\beta))$ is the garbled circuit analogous to GC above, computed by AG_1 in the multi-party protocol;
- $decode_j$ is the decoding information for the output of garbled circuit GC_j;
- for $j > 1$, $gc_j = gc(\langle id_j, input_j \rangle, transl(decode_{j-1}))$ is the garbled circuit computed by Ag_j, with input $\langle id_j, input_j \rangle$ hardwired to it, and translation information $transl(decode_{j-1})$ for translating the garbled output of the cascade of circuits computed by agencies Ag_1 through Ag_{j-1} to a garbled input for gc_j;
- GC_j is gc_j concatenated to GC_{j-1};
- $eval(GC_n, decode_n, \beta)$ denotes evaluation of circuit GC_n with decoding information $decode_n$ to interpret the output and additional input β to complete the OT.