Hyperproperties for Safe and Secure RFID Systems

Ludovico Fusco[1,*,†], Alessandro Aldini[1,*,†]

[1]Department of Pure and Applied Sciences, University of Urbino, Italy

Abstract

Although there have been many contributions to the rigorous description and verification of RFID-based systems and their safety and security properties, there has yet to be much progress toward an explicit formalization of information flow policies for such systems in terms of hyperproperties. In this paper, we introduce three classes of hyperproperties related to the analysis of anti-collision protocols for RFID tags: hyper-reachability, hyper-adaptivity, and generalized non-interference. As a modeling framework, we employ an event-based model (suitable for representing a large portion of existing RFID systems, both with passive and battery-powered tags) featuring a component-oriented notion of state and allowing us to express hyperproperties in terms of event satisfaction by component configurations. For each hyperproperty, we provide a formalization *à la* Clarkson-Schneider and a hyperlogic characterization. We also propose some insights about decidability issues.

Keywords

Hyperproperties, RFID systems, Anti-collision protocols, Hyperlogics

1. Introduction

Since its official patenting in the 1980s, the Radio-Frequency Identification (RFID) technology has been experiencing relentless advancement due to its extreme versatility and usability in a wide range of contexts such as access control, logistics, retail, and supply chain management [1]. Moreover, as an enabling technology for the IoT computing paradigm, RFID today underpins new systems and protocols for object identification/data acquisition in smart environments [2]. At the same time, RFID-based systems are proving vulnerable to attacks conducted against RFID technology, such as cloning, replaying, relaying, and even backend attacks that exploit RFID vulnerabilities to inject commands to the backend of the system, including middleware and database management systems [3, 4, 5].

Despite a large number of contributions to the formal description and verification of RFID-based systems and protocols, to the best of our knowledge, no foundational framework has yet been proposed for the explicit formalization of information flow policies for RFID systems in terms of hyperproperties [6]. The need for studies in this direction has emerged especially, though not always explicitly, in the area of information security (see, e.g., [7, 8, 9, 10, 11]).

This paper aims to be a first step toward a taxonomy of hyperproperties for RFID systems, laying the foundation for a general framework for their formalization. To this end, we introduce a low-level, trace-based model suitable for representing a large portion of existing RFID systems (both with passive and battery-powered tags) implementing tree protocols for tag collision arbitration [12]. Our model features a component-oriented, event-based notion of state allowing us to express hyperproperties in terms of event satisfaction by component configurations. Our model assumes a prior high-level specification in terms of Kripke structures, automata, or labeled transition systems, which is not provided in this paper. The notions of state and trace are therefore intended to derive from such a specification. Moreover, system executions are assumed to be synchronous and discrete-time.

Within this framework, we introduce three classes of hyperproperties for the analysis of tree-based anti-collision protocols for RFID tags, and we discuss the safety and security conditions they allow us to

ITASEC 2024: The Italian Conference on CyberSecurity, April 08–12, 2024, Salerno, Italy

*Corresponding author.

†These authors contributed equally.

l.fusco2@campus.uniurb.it (L. Fusco); alessandro.aldini@uniurb.it (A. Aldini)

https://www.uniurb.it/persone/alessandro-aldini (A. Aldini)

0009-0001-5844-2894 (L. Fusco); 0000-0002-7250-5011 (A. Aldini)

investigate. The three classes are hyper-reachability, hyper-adaptivity, and generalized non-interference. While generalized non-interference is an instance of the general template introduced in [13], hyper-reachability and hyper-adaptivity are novel and specifically designed for RFID systems. For each class, we provide a classical definition in the style of [6] and a formalization in a suitable hyperlogic [14]. It turns out that hyper-reachability and generalized non-interference are expressible in HyperLTL, while hyper-adaptivity requires the first-order logic $\mathrm{FO}[<, E]$. On the basis of the quantifier prefix of each formula obtained, we provide information on decidability and complexity results with respect to the problems of satisfiability and model checking in the combined input Kripke structure/HyperLTL formula (the latter problem in the cases of hyper-reachability and generalized non-interference only).

The paper is organized as follows. In Section 2, we provide some background information about RFID technology and tree-based anti-collision protocols for RFID tags. In Sections 3 and 4, we present our abstract model of an RFID system and set up the machinery for hyperproperty formalization. In Section 5, we propose our three classes of hyperproperties for RFID verification. In Section 6, we provide some closing remarks and indicate possible directions for future research.

2. Background

2.1. RFID systems

An RFID system is made up of three main components: a controller, a reader, and a batch of tags. A tag is a transponder consisting of a microchip connected to an antenna. The microchip can store and process (a small amount of) data, which can be transmitted by the antenna in the form of radio signals. RFID tags can be either passive, semi-passive or active. Passive tags do not have an embedded power source. Semi-passive and active tags, on the other hand, are provided with an internal battery, but differ in an important respect: in the former, the battery powers the microchip and possibly some additional on-board component (e.g., a sensor), but does not supply energy for signal transmission; in the latter, it performs both functions. The reader is a device (also equipped with its own antenna) capable of uniquely identifying, tracking or locating tags by generating an electromagnetic field. The controller is a backend workstation connected to the reader via a network interface (either wired or wireless). Depending on the particular infrastructure the RFID system is integrated into, the controller carries out a number of different tasks. First and foremost, it is responsible for collecting and somehow processing the raw data captured by the reader.

An RFID system is activated each time the reader emits a data request signal through its antenna. In the case of passive and semi-passive systems, if a tag is within the reader's interrogation zone, the signal transmitted by the reader is picked up by the tag's antenna and converted into electrical energy for the microchip's circuitry. At this point, the tag transmits the requested information to the reader via load/backscatter modulation. In the case of active systems, the tag responds to the reader using its own power source.

2.2. The symmetry problem

In this paper, we only consider systems where the communication between the reader and the tags takes place on a single multiple access channel. In such a context, tag-to-reader collisions [15] can occur whenever two or more tags respond to a data request from the reader at the same time. Indeed, simultaneous reply transmissions interfere with each other and, as a result, the reader cannot distinguish between the tags. This situation is commonly referred to as the *symmetry problem*. As for collision arbitration, we assume that the system implements a well-known tree algorithm based on random bit extraction by the tags and identification via binary address by the reader. Originally developed in the broader context of conflict resolution techniques in shared communication channels, this protocol was independently introduced and studied in [16, 17, 18]. Over time, it has undergone several modifications and optimizations, many of which have been successfully applied to the RFID context [12].

Table 1
Input and output ports of S.

C		R		tag_k	
Input	Output	Input	Output	Input	Output
$\mathtt{data}_{in}$	$\mathtt{p\_data}_{out}$	$\mathtt{reply}_{in}$	$\mathtt{data\_request}_{out}$ $\mathtt{address}_{out}$ $\mathtt{feedback}_{out}$ $\mathtt{data}_{out}$	$\mathtt{data\_request}^k_{in}$ $\mathtt{address}^k_{in}$ $\mathtt{feedback}^k_{in}$	$\mathtt{reply}^k_{out}$ $\mathtt{r\_bit}^k_{out}$

In the event of a collision, the protocol states that the reader sends negative feedback to all tags in its interrogation zone, asking them to extract a random bit. After that, the reader interrogates all tags that extracted, say, 0. If a collision still occurs, then this splitting procedure is iterated until the reader successfully communicates with every tag in the first group. The procedure is then repeated for tags that initially extracted 1. At the end of the protocol, each tag has its own unique identifying bitstring (address) consisting of the sequence of outcomes of all its extractions.

3. Model description

In this section, we introduce our model of an RFID system and specify its essential architecture by detailing the input and output ports relevant to our analysis. On this basis, we define the notions of event, state, and trace.

3.1. Components and interfaces

We represent an RFID system as a triple $\mathsf{S} = \langle \mathsf{C}, \mathsf{R}, \mathsf{T} \rangle$, where C is the backend controller, R is the reader, and $\mathsf{T} = \{\mathsf{tag}_k\}_{k \in K}$ is a batch of tags indexed over a finite integer set K of cardinality $|K| > 1$. We denote input and output ports of S using the notation $\mathtt{port}_c$, where $\mathtt{port}$ is a port name and $c \in \{in, out\}$ is a port type (see Table 1). Each component tag_k is equipped with an input port $\mathtt{data\_request}^k_{in}$ to receive instructions from R and an output port $\mathtt{reply}^k_{out}$ to send back information. Clearly, R is provided with the corresponding ports $\mathtt{data\_request}_{out}$ and $\mathtt{reply}_{in}$. The raw data that the reader acquires from the tags are sent to C via $\mathtt{data}_{out}$ and received at $\mathtt{data}_{in}$. Once the data have been processed, the controller outputs them in their final form through the port $\mathtt{p\_data}_{out}$. Since the total number of tags is $|K|$, a tag collision of multiplicity at most $|K|$ can occur every time the reader sends a request. In each round of communication, the reader assesses whether the transmissions were successful or not and sends feedback to the tags in its interrogation zone via the port $\mathtt{feedback}_{out}$. In particular, $\mathtt{feedback}_{out}$ returns the values 0 (*idle*), 1 (*success*) and $2+$ (*collision*). Feedback is received by the tags at $\mathtt{feedback}^k_{in}$. In the case of a collision, the tags and the reader enable, respectively, the ports $\mathtt{r\_bit}^k_{out}$ (used for random bit extraction) and $\mathtt{address}_{out}$ (used for tag identification and transmission attempt authorization during the execution of the anti-collision protocol). The address bitstring and the random bit are sent to the input port $\mathtt{address}^k_{in}$ of the tag, which compares the last bit of the address with the extracted random bit, possibly authorizing a new data transmission via $\mathtt{reply}^k_{out}$ (the information flows among the ports are depicted in Figure 1).

3.2. Events, states and traces

Our model is based on the primitive notion of *event*. An event represents the activity state of a single port of the system. In our framework, events are atomic entities, so they cannot be combined into more "complex" events, nor can they describe the activity of more than one port. We use ad-hoc notations for three different types of events. The first one is $\mathtt{port}_c : b$, where $b \in \{0, 1\}$ is a Boolean value. We write $\mathtt{port}_c : b$ to express the fact that, depending on c, an input or an output value for $\mathtt{port}_c$ is present ($b = 1$) or absent ($b = 0$) and, accordingly, the information flows or does not flow through the port. Sometimes, we will also need to make explicit *what* input or output values are present for a certain

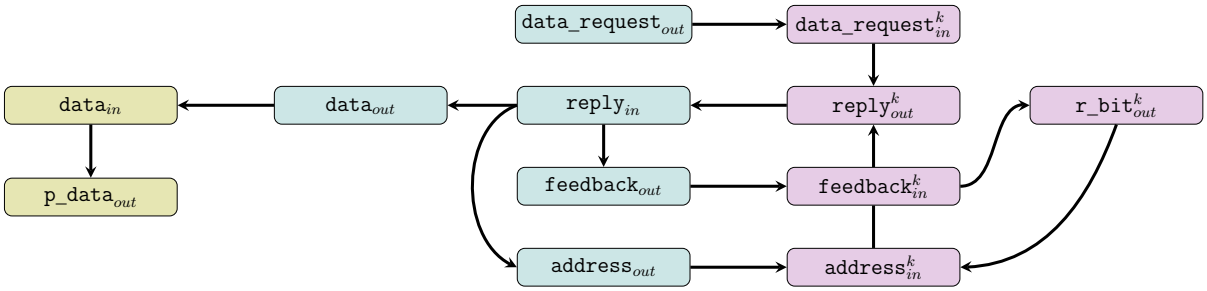

Figure 1: Dependencies between input and output ports of controller, reader and tags.

port. So we write $\texttt{port}_{in} \triangleleft X$ and $\texttt{port}_{out} \triangleright X$ to mean that $\texttt{port}_{in}$ is taking the value X as input and $\texttt{port}_{out}$ is returning the value X as output, respectively.

Events are the building blocks of the notions of configuration and state. Let X be a component of S. An *event set* (*e-set*) for X is any set of events related to the ports of X. We say that an e-set for X is *well-defined* if it contains exactly one event $\texttt{port}_c : b$ for each port of X and, whenever $b = 1$, it possibly contains an event making explicit the value present for $\texttt{port}_c$. A *configuration* of X is a well-defined e-set for X. A *state* of S is a tuple of configurations of the form $\sigma = \langle E_{\mathsf{C}}, E_{\mathsf{R}}, E_{\mathsf{tag}_1}, \ldots, E_{\mathsf{tag}_{|K|}} \rangle$. For a state σ and a component X, we write $\sigma.E_{\mathsf{X}}$ to denote the configuration E_{X} at σ. The *state space* of S is denoted by Σ_{S}. A *trace* is an ω-word τ over Σ_{S}. We denote by T_{S} the set of execution traces of S.

Finally, we define event satisfaction. We say that an event e is *satisfied* by the configuration of X at state σ (notation: $\sigma.E_{\mathsf{X}} \vDash e$) whenever $e \in \sigma.E_{\mathsf{X}}$.

3.3. The anti-collision protocol

Following [12], we present below two pseudocodes illustrating the anti-collision protocol from the perspective of both the reader (Algorithm 1) and a generic tag (Algorithm 2). The instructions are formulated in accordance with our modeling framework.

Algorithm 1 The protocol run at R

```
1  a = [ ];
2  end = 0;
3  while end == 0:
4      data_request_out ▷ request
5      address_out ▷ a
6      if reply_in ◁ × :  // collision
7          feedback_out ▷ 2+
8          a.push(0)
9      else if reply_in ◁ value:
10         feedback_out ▷ 1
11         while a.read() == 1:
12             a.pop()
13         if a == [ ]:
14             end = 1
15         else:
16             a.pop()
17             a.push(1)
18     else: feedback_out ▷ 0
```

Algorithm 2 The protocol run at tag_k

```
1  a_k = [ ];
2  end = 0;
3  while end == 0;
4      data_request_in^k ◁ request
5      address_in^k ◁ a
6      if a == a_k:
7          reply_out^k ▷ value
8          if feedback_in^k ◁ 2+:
9              r_bit_out^k ▷ b
10             a_k.push(b)
11         else if feedback_in^k ◁ 1:
12             end = 1
```

4. Logical framework

A standard approach for describing and reasoning about the behavior of a computational system is to represent it as the set of all its execution traces, i.e., the sequences of states corresponding to the computations that the system performs. This method allows to model many policies satisfied by the system in terms of trace properties (in this perspective, a policy is nothing more than the set of execution traces complying with it), offering a natural framework for a classification based on the safety-liveness dichotomy [19]. However, in this context, one cannot specify those policies whose modeling and verification must take into account the interaction between a system and its environment. Non-interference [20] and many other fundamental information flow security policies [21] typically fall into this category.

In their celebrated paper [6], Clarkson and Schneider introduced *hyperproperties*, a powerful formalism for expressing such policies. Hyperproperties specify conditions under which the behavior of a system in certain runs does or does not depend on the characteristics of certain alternative executions. It is precisely because of this major difference from trace properties that hyperproperties must be formally modeled as *properties of systems*. So, if a trace property is a set of execution traces, a hyperproperty is a set of trace properties. Accordingly, if the definitions of nontrivial trace properties require quantification on instants of time, those of hyperproperties require an additional level of quantification on traces in order to impose conditions on different executions and to relate them to each other.

In the following, we show how to formalize hyperproperties for RFID systems in light of our event-based model.

4.1. Hyperproperties

We start by discussing the modeling of a classical trace property for S, namely reachability, prescribing that whenever the reader sends a request to the tags, after some time the k-th tag sends back a successful response. This is an example of liveness policy, and as such it is expressible in Linear Temporal Logic (LTL) [22]. Indeed this policy corresponds to the trace property $\mathbf{P}_{\mathsf{S}}[k] \subseteq \Sigma_{\mathsf{S}}^{\omega}$ consisting of all traces satisfying the LTL formula $\Box(\varphi \rightarrow \Diamond\psi)$ where:

$$\varphi := [E_{\mathsf{R}} \vDash \texttt{data\_request}_{out} : 1], \text{ and}$$
$$\psi := [E_{\mathsf{tag}_k} \vDash \texttt{reply}^{k}_{out} : 1] \wedge \bigcirc\bigcirc[E_{\mathsf{R}} \vDash \texttt{feedback}_{out} \triangleright 1].$$

Note that we assume the reader returns feedback two states after the tag's response. Let AP be a set of atomic propositions. Observe that the standard semantic clause for $a \in AP$ in LTL is formulated as $\tau, i \models a \Leftrightarrow a \in \tau[i]$, where τ is a trace and $i \in \mathbb{N}$. In our case, every such an a is the counterpart of a satisfaction statement of the form $\tau[i].E_{\mathsf{X}} \vDash e$. Since states and traces become explicit only at the semantic level, we unfold the internal structure of atomic propositions using the notation $[E_{\mathsf{X}} \vDash e]$. Thus, according to our event-based framework, we obtain

$$\tau, i \models [E_{\mathsf{X}} \vDash e] \Leftrightarrow e \in \tau[i].E_{\mathsf{X}} \Leftrightarrow \tau[i].E_{\mathsf{X}} \vDash e.$$

By Kamp's Theorem [23] (see also [24, 25, 26]), we know that LTL is expressively equivalent to FO[$<$], the fragment of first-order logic (with equality) over the class of Dedekind complete linearly ordered structures with monadic predicates $\{P_a\}_{a \in AP}$ (in this paper, our reference model is clearly built over $\mathbb{N}$). With a slight abuse of notation, we write $[E_{\mathsf{X}} \vDash e](x)$ for the atomic open FO[$<$] formula corresponding to $[E_{\mathsf{X}} \vDash e]$ in LTL. We can therefore redefine $\mathbf{P}_{\mathsf{S}}[k]$ as the class of all traces $\tau \in \Sigma_{\mathsf{S}}^{\omega}$ satisfying the FO[$<$] formula $\forall x.(\varphi(x) \rightarrow \exists y > x.\psi(y))$, where:

$$\varphi(x) := [E_{\mathsf{R}} \vDash \texttt{data\_request}_{out} : 1](x), \text{ and}$$
$$\psi(y) := [E_{\mathsf{tag}_k} \vDash \texttt{reply}^{k}_{out} : 1](y) \wedge [E_{\mathsf{R}} \vDash \texttt{feedback}_{out} \triangleright 1](y+2).$$

In [6], Clarkson and Schneider's formalization assumes, though does not always explicitly use, a two-sorted first-order notation allowing quantification both over traces and over instants of time. In this

setting, elementary facts about system behaviors are expressed by binary predicates of the form $P(\tau, i)$. We can think of each of these predicates as being associated with an atomic proposition $a \in AP$. So, in our context, for $a = [E_{\mathsf{X}} \vDash e]$, we have $P_a(\tau, i) := \tau[i].E_{\mathsf{X}} \vDash e$.

Going back to the previous example, suppose we want to model an information flow policy based on $\mathbf{P}_{\mathsf{S}}[k]$ requiring that for every execution in which the reader *globally* fails to communicate with the k-th tag, there is at least one in which communication is successful. For $n \in \{1, 2\}$, let us define:

$$\begin{aligned}
\varphi(\tau_n, i_n) &:= \tau_n[i_n].E_{\mathsf{R}} \vDash \texttt{data_request}_{out} : 1 \\
\psi(\tau_n, j_n) &:= (\tau_n[j_n].E_{\mathsf{tag}_k} \vDash \texttt{reply}^k_{out} : 1) \wedge (\tau_n[j_n + 2].E_{\mathsf{R}} \vDash \texttt{feedback}_{out} \triangleright 1) \\
\chi^n_{fail} &:= \forall i_n.\exists j_n > i_n.(\varphi(\tau_n, i_n) \rightarrow \neg\psi(\tau_n, j_n)) \\
\chi^n_{succ} &:= \forall i_n.\exists j_n > i_n.(\varphi(\tau_n, i_n) \rightarrow \psi(\tau_n, j_n))
\end{aligned}$$

So we can define a hyperproperty $\mathbf{HP}_{\mathsf{S}}[k]$ consisting of all $T \subseteq \Sigma^{\omega}_{\mathsf{S}}$ satisfying:

$$\forall\tau_1.\exists\tau_2.\big(\chi^1_{fail} \rightarrow \chi^2_{succ}\big).$$

The hyperproperties we introduce in Section 5 are defined in this way.

4.2. Hyperlogics

In recent years, much effort has been devoted to the development of logics for verifying hyperproperties for both linear and branching time systems. The wide variety of formalisms introduced (known as *hyperlogics*) has led to the construction of an expressiveness hierarchy [14] that is constantly being enriched with new elements. In this paper, we formalize our hyperproperties using the well-known linear time hyperlogics HyperLTL and FO$[<, E]$.

HyperLTL [27] extends LTL with an additional layer of syntax enabling quantification over trace variables from a denumerable set $X = \{\pi_1, \pi_2, \dots\}$. Further, atomic propositions are annotated with the trace variables with respect to which they are to be evaluated. Well-formed formulas are defined by the following grammar:

$$\begin{aligned}
\psi &::= \varphi \mid \forall\pi.\psi \mid \exists\pi.\psi \\
\varphi &::= a_\pi \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi \,\mathcal{U}\, \varphi
\end{aligned}$$

Evaluation of HyperLTL formulas is defined with respect to a set of traces $T \subseteq \wp(AP)^{\omega}$ and a (partial) trace assignment $\Pi : X \rightarrow T$. In the following list, we denote by $\Pi[\pi \mapsto \tau]$ the assignment that coincides with Π except for the trace τ assigned to π.

$$\begin{aligned}
&\Pi, i \models_T a_\pi && \Leftrightarrow\ a \in \Pi(\pi)[i] \\
&\Pi, i \models_T \neg\varphi && \Leftrightarrow\ \Pi, i \not\models_T \varphi \\
&\Pi, i \models_T \varphi_1 \wedge \varphi_2 && \Leftrightarrow\ \Pi, i \models_T \varphi_1 \text{ and } \Pi, i \models_T \varphi_2 \\
&\Pi, i \models_T \bigcirc\varphi && \Leftrightarrow\ \Pi, i + 1 \models_T \varphi \\
&\Pi, i \models_T \varphi_1 \,\mathcal{U}\, \varphi_2 && \Leftrightarrow\ \exists j \geq i.\Pi, j \models_T \varphi_2 \text{ and } \forall k \in [i, j).\Pi, k \models_T \varphi_1 \\
&\Pi, i \models_T \forall\pi.\varphi && \Leftrightarrow\ \forall\tau \in T.\Pi[\pi \mapsto \tau], i \models_T \varphi \\
&\Pi, i \models_T \exists\pi.\varphi && \Leftrightarrow\ \exists\tau \in T.\Pi[\pi \mapsto \tau], i \models_T \varphi
\end{aligned}$$

The temporal modalities $\Diamond$ (eventually) and $\Box$ (globally) are defined in the usual way, with $\Diamond\varphi := \mathit{true}\,\mathcal{U}\,\varphi$ and $\Box\varphi := \neg\Diamond\neg\varphi$. Note that, as regards our case study, the semantic clause for atomic propositions becomes:

$$\Pi, i \models_T [E_{\mathsf{X}} \vDash e]_\pi \Leftrightarrow e \in \Pi(\pi)[i].E_{\mathsf{X}} \Leftrightarrow \Pi(\pi)[i].E_{\mathsf{X}} \vDash e.$$

Although HyperLTL allows for the description of a wide variety of hyperproperties, this logic lacks sufficient syntactic resources to express equality/non-equality statements between instants of time.

This problem does not arise in the first-order logic FO$[<, E]$ [28], which is obtained by extending the syntax of FO$[<]$ with the addition of a new binary predicate symbol E that expresses equality between instants of time (with respect to either the same or different traces). Starting from a denumerable set of variables $X = \{x_1, x_2, \dots\}$, well-formed FO$[<, E]$ formulas are given by the following grammar:

$$\begin{aligned} \psi &::= \varphi \mid \neg\psi \mid \psi_1 \wedge \psi_2 \mid \forall x.\psi \mid \exists x.\psi \\ \varphi &::= P_a(x) \mid x < y \mid x = y \mid E(x, y) \end{aligned}$$

The intended models of FO$[<, E]$ are structures with universe $T \times \mathbb{N}$, where $T \subseteq \wp(AP)^\omega$ is a set of traces. For $\langle \tau, i \rangle \in T \times \mathbb{N}$ and $l \in \{1, 2\}$, we denote by $proj_l(\langle \tau, i \rangle)$ the projection on the l-th component of $\langle \tau, i \rangle$. The semantics of FO$[<, E]$ is defined with respect to (partial) pair assignments $V : X \to T \times \mathbb{N}$, according to the following clauses:

$$\begin{aligned} V \models_T P_a(x) &\Leftrightarrow a \in \tau[i], \text{ where } \tau = proj_1(V(x)) \text{ and } i = proj_2(V(x)) \\ V \models_T x < y &\Leftrightarrow proj_1(V(x)) = proj_1(V(y)) \text{ and } proj_2(V(x)) < proj_2(V(y)) \\ V \models_T x = y &\Leftrightarrow V(x) = V(y) \\ V \models_T E(x, y) &\Leftrightarrow proj_2(V(x)) = proj_2(V(y)) \\ V \models_T \neg\psi &\Leftrightarrow V \not\models_T \psi \\ V \models_T \psi_1 \wedge \psi_2 &\Leftrightarrow V \models_T \psi_1 \text{ and } V \models_T \psi_2 \\ V \models_T \forall x.\psi &\Leftrightarrow \forall \tau \in T. \forall i \in \mathbb{N}. V[x \mapsto \langle \tau, i \rangle] \models_T \psi \\ V \models_T \exists x.\psi &\Leftrightarrow \exists \tau \in T. \exists i \in \mathbb{N}. V[x \mapsto \langle \tau, i \rangle] \models_T \psi \end{aligned}$$

FO$[<, E]$ has been shown to be strictly more expressive than HyperLTL [28, Thm. 8]. Nevertheless, an analog of Kamp's theorem for HyperLTL has been proved for a fragment of FO$[<, E]$ [28, Thm. 9]. Finally, recall that, with respect to sets of infinite traces, FO$[<, E]$ is expressively equivalent to the recently introduced Hypertrace Logic [29], whose syntax incorporates and extends the two-sorted notation of Clarkson and Schneider. All definitions of our hyperproperties can thus be seen as formalizations in this hyperlogic.

5. Hyperproperties for RFID systems

In this section, we introduce three hyperproperty templates for RFID systems: hyper-reachability, hyper-adaptivity, and generalized non-interference. This choice is not accidental. In fact, all three policies can be related to both safety and security contexts, so that compliance with them represents a prerequisite for the optimal operation of any system that fits the model presented in Section 3.

5.1. Hyper-reachability

Among the policies our system has to comply with in order to have full functionality, there should be one prescribing that, for every execution where the reader is able to successfully communicate with only m tags (for $m < |K|$), there exist other executions where it manages to communicate with all the remaining $|K| - m$ tags. However, since the tag-to-reader data transmissions can be compromised by several problems that the system itself cannot solve, this policy is too strong to be enforced in real cases. We thus propose a weaker version thereof by relaxing the condition on the existence of the alternative runs. In particular, our policy prescribes that there exists at least one execution where the reader successfully communicates with at least one nonempty subset of the remaining $|K| - m$ tags. Since we are reasoning in terms of pairs of disjoint sets of tags, we are actually defining a class of hyperproperties for S, each one defined with respect to a two-element partition of K.

Definition 1 (Hyper-reachability)**.** Let $\Pi_2(K)$ be the set of all the two-element partitions of K. For $\mathcal{P} \in \Pi_2(K)$ with blocks P, Q, the *hyper-reachability property* for S with respect to $\mathcal{P}$ is the set $\mathbf{HR}_{\mathsf{S}}[\mathcal{P}]$

of all $T \subseteq \Sigma_{\mathsf{S}}^{\omega}$ satisfying:

$$\forall \tau_1 . \exists \tau_2 . \left(\left(\bigwedge_{p \in P} \exists i . \varphi_p(\tau_1, i) \wedge \bigwedge_{q \in Q} \neg \exists i' . \varphi_q(\tau_1, i') \right) \rightarrow \bigvee_{\substack{U \subseteq Q \\ U \neq \emptyset}} \bigwedge_{u \in U} \exists j . \varphi_u(\tau_2, j) \right)$$

where, for $n \in \{1, 2\}$, $k \in K$, and $i \in \mathbb{N}$:

$$\varphi_k(\tau_n, i) := (\tau_n[i] . E_{\mathsf{tag}_k} \vDash \mathtt{reply}^k_{out} : 1) \wedge (\tau_n[i+2] . E_{\mathsf{R}} \vDash \mathtt{feedback}_{out} \triangleright 1).$$

We say that S satisfies the *weak* (resp., the *strong*) *hyper-reachability property* if $T_{\mathsf{S}} \in \bigcup_{\mathcal{P} \in \Pi_2(K)} \mathbf{HR}_{\mathsf{S}}[\mathcal{P}]$ (resp., $T_{\mathsf{S}} \in \bigcap_{\mathcal{P} \in \Pi_2(K)} \mathbf{HR}_{\mathsf{S}}[\mathcal{P}]$).

Hyper-reachability admits the following formalization in HyperLTL:

$$\forall \pi_1 . \exists \pi_2 . \left(\left(\bigwedge_{p \in P} \Diamond \varphi_p(\pi_1) \wedge \bigwedge_{q \in Q} \neg \Diamond \varphi_q(\pi_1) \right) \rightarrow \bigvee_{\substack{U \subseteq Q \\ U \neq \emptyset}} \bigwedge_{u \in U} \Diamond \varphi_u(\pi_2) \right)$$

where, for $n \in \{1, 2\}$ and $k \in K$:

$$\varphi_k(\pi_n) := [E_{\mathsf{tag}_k} \vDash \mathtt{reply}^k_{out} : 1]_{\pi_n} \wedge \bigcirc\bigcirc [E_{\mathsf{R}} \vDash \mathtt{feedback}_{out} \triangleright 1]_{\pi_n}.$$

It is well known that the satisfiability problem for the class of HyperLTL formulas with quantifier prefix $\forall\exists$ is undecidable [30, Thm. 14]. The model checking problem for the same class is instead EXPSPACE-complete [27, 31].

In summary, hyper-reachability ensures that no environmental factor can prevent a given set of tags from communicating with the reader *in any possible system execution*. To further clarify this point, let us take a concrete example. In the RFID context, *read reliability* is defined as the probability that a tag is recognized when it is placed within the reader's interrogation zone. Experimental results have shown that reliability depends, among other things, on the orientation of the tag's antenna relative to the reader's one [32]. Thus, an improper spatial arrangement of tags, either due to random factors or malicious tampering, could compromise the success of some transmissions by lowering the read reliability value for some tags. An RFID system satisfying hyper-reachability must therefore be part of an infrastructure that is appropriately designed to prevent the above problem from occurring with respect to the same tags in every possible situation. Hyper-reachability is therefore a policy that must be enforced at the physical layer.

5.2. Hyper-adaptivity

Very often, the environmental conditions in which the reader queries tags are not static. The spatial layout of components and the order in which transmissions occur can vary from execution to execution, potentially resulting in different data acquisitions by the reader and processing outcomes by the controller. We introduce an adaptivity policy for S stating that, for any system execution in which the controller outputs a value v (assumed to be a real number), there are m distinct alternative executions in each of which the reader successfully makes n distinct acquisitions and the controller outputs an approximated value of the form $v \pm \varepsilon$, with ε a tolerance parameter. This is an adaptive policy because its satisfaction would show that the system can adapt to a dynamic context while still ensuring mutually consistent results. Clearly, m, n, and ε represent indices of robustness: the larger m, n and the smaller ε, the more robust the system.

Definition 2 (Hyper-adaptivity)**.** For $m, n \in \mathbb{N}$ and $\varepsilon \in \mathbb{R}$, the *hyper-adaptivity property* for S with respect to m, n, and ε is the set $\mathbf{HA}_{\mathsf{S}}[m, n, \varepsilon]$ of all $T \subseteq \Sigma_{\mathsf{S}}^{\omega}$ satisfying:

$$\forall\tau.\exists_{j=1}^{m}\tau_j.\bigwedge_{j\neq j'}\tau_j \neq \tau_{j'} \wedge \forall i.\left(\varphi(\tau,i) \rightarrow \bigwedge_{j=1}^{m}\left(\exists h^j.\psi_j(\tau_j,h^j) \wedge \exists_{l=1}^{n} i_l^j < h^j.\bigwedge_{l\neq l'} i_l^j \neq i_{l'}^j \wedge \bigwedge_{l=1}^{n}\chi_j(\tau_j,i_l^j)\right)\right)$$

where $\varphi(\tau, i)$, $\psi_j(\tau_j, h^j)$, and $\chi_j(\tau_j, i_l^j)$ are defined as follows:

$$\begin{aligned}\varphi(\tau,i) &:= \tau[i].E_{\mathsf{C}} \vDash \texttt{p\_data}_{out} \triangleright v,\\ \psi_j(\tau_j,h^j) &:= \tau_j[h^j].E_{\mathsf{C}} \vDash \texttt{p\_data}_{out} \triangleright v \pm \varepsilon,\\ \chi_j(\tau_j,i_l^j) &:= \tau_j[i_l^j].E_{\mathsf{R}} \vDash \texttt{feedback}_{out} \triangleright 1.\end{aligned}$$

A hyperlogic characterization of $\mathbf{HA}_{\mathsf{S}}[m, n, \varepsilon]$ can be provided by the following $\mathrm{FO}[<, E]$ formula (where $\varphi(\langle\tau, i\rangle)$, $\psi_j(\langle\tau_j, h^j\rangle)$, and $\chi_j(\langle\tau_j, i_l^j\rangle)$ are defined according to the conventions adopted in Section 4):

$$\begin{aligned}&\forall\langle\tau,i\rangle.\exists_{j=1}^{m}\langle\tau_j,i^j\rangle.\exists\langle\tau_j,h^j\rangle.\exists_{l=1}^{n}\langle\tau_j,i_l^j\rangle.\bigwedge_{j\neq j'}\langle\tau_j,i^j\rangle \neq \langle\tau_{j'},i^{j'}\rangle \wedge \bigwedge_{l=1}^{n}\langle\tau_j,i_l^j\rangle < \langle\tau_j,h_l^j\rangle\\ &\wedge \bigwedge_{l\neq l'}\neg E\left(\langle\tau_j,i_l^j\rangle,\langle\tau_j,i_{l'}^j\rangle\right) \wedge \left(\varphi(\langle\tau,i\rangle) \rightarrow \bigwedge_{j=1}^{m}\left(\psi_j(\langle\tau_j,h^j\rangle) \wedge \bigwedge_{l=1}^{n}\chi_j(\langle\tau_j,i_l^j\rangle)\right)\right).\end{aligned}$$

It can be easily checked that this formula belongs to the Ackermann class with equality, i.e., the fragment $[\exists^*\forall\exists^*, all]_=$ of first order logic with equality, for which the satisfiability problem has been shown to be decidable with complexity $\textsc{ntime}(2^{cn/\log n})$, for c some constant [33, Cor. 6.3.30].

Hyper-adaptivity is designed for all those infrastructures where tags transmit data acquired from some sensing device. Consider an RFID system where each tag has an on-board sensor for temperature measurement. Suppose that the reader collects temperature data and sends them to the controller, which returns an arithmetic mean of the received values. In such a scenario, hyper-adaptivity ensures that, for any system execution where the controller computes a mean v at the end of a communication round, there are *sufficient* (at least m) and *relevant* distinct alternative runs (each with n distinct temperature acquisitions) where the controller obtains the same mean up to a tolerance parameter ε. Suppose, for instance, that some tags are temporarily disabled due to a jamming attack or a passive interference, or that after a cloning attack on some tags, the counterfeit tags transmit incorrect data. In both cases, the mean v may be either incorrect or based on too few data acquisitions. Therefore, the sensitivity analysis with respect to the choices of the indices m, n, and ε is helpful to provide insights about the impact of such kinds of situations.

5.3. Generalized non-interference

Non-interference provides one of the most important families of information flow security policies. Developed in the context of multilevel security, it was introduced by Goguen and Meseguer [20] in a deterministic setting and generalized to nondeterministic systems by McCullough [34] and McLean [13]. Subsequently, non-interference has been widely studied in many different frameworks, such as, e.g., the semantics of concurrent programs [35], language-based static analysis [36], abstract interpretation [37], property compositionality in process algebra [38], and reversible computing [39].

Here we show how a generalized notion of non-interference can be adapted to a specific condition of interest (with respect to both safety and security aspects) for RFID systems. During the execution of an anti-collision protocol, random bit extractions by tags are events that should under no circumstances compromise the transmission of data to the reader. We express this policy as a generalized non-interference hyperproperty [13] parameterized with respect to a single component tag_k. Our policy prescribes that, for any two traces, there exists a third interleaved trace that is globally equivalent to

the first with respect to random extractions by tag_k and to the second with respect to the successful detection of tag_k by the reader. This condition guarantees that, at every possible instant of time, successful detection is independent of random extraction.

Definition 3 (Generalized non-interference)**.** For $k \in K$, the *generalized non-interference property* for S with respect to k is the set $\mathbf{GNI}_{\mathsf{S}}[k]$ of all $T \subseteq \Sigma_{\mathsf{S}}^{\omega}$ satisfying:

$$\forall\tau_1.\forall\tau_2.\exists\tau_3.\forall i.(\varphi(\tau_1, i) \leftrightarrow \varphi(\tau_3, i) \wedge \psi(\tau_2, i) \leftrightarrow \psi(\tau_3, i))$$

where, for $n \in \{1, 2, 3\}$ and $i \in \mathbb{N}$, $\varphi(\tau_n, i) := \tau_n[i].E_{\mathsf{tag}_k} \vDash \mathtt{r\_bit}_{out} : 1$ and

$$\psi(\tau_n, i) := \tau_n[i].E_{\mathsf{tag}_k} \vDash \mathtt{reply}^k_{out} : 1 \wedge \tau_n[i+2].E_{\mathsf{R}} \vDash \mathtt{feedback}_{out} \triangleright 1.$$

We say that S satisfies the *strong generalized non-interference property* if $T_{\mathsf{S}} \in \bigcap_{k \in K} \mathbf{GNI}_{\mathsf{S}}[k]$.

$\mathbf{GNI}_{\mathsf{S}}[k]$ can be expressed in HyperLTL:

$$\forall\pi_1.\forall\pi_2.\exists\pi_3.\Box(\varphi(\pi_1) \leftrightarrow \varphi(\pi_3) \wedge \psi(\pi_2) \leftrightarrow \psi(\pi_3))$$

where, for $n \in \{1, 2, 3\}$, $\varphi(\pi_n) := [E_{\mathsf{tag}_k} \vDash \mathtt{r\_bit}_{out} : 1]_{\pi_n}$ and:

$$\psi(\pi_n) := [E_{\mathsf{tag}_k} \vDash \mathtt{reply}^k_{out} : 1]_{\pi_n} \wedge \bigcirc\bigcirc[E_{\mathsf{R}} \vDash \mathtt{feedback}_{out} \triangleright 1]_{\pi_n}.$$

Decidability results are the same as in the hyper-reachability case.

Notice that $\mathbf{GNI}_{\mathsf{S}}[k]$ describes a safety condition as its satisfaction is a prerequisite for the correct execution of collision arbitration. However, it can also be used to formalize a security property to verify whether the system mitigates attacks intended to alter the random extraction and invalidate the protocol. While this is just an instance of generalized non-interference, an analogous template can formalize several security aspects of RFID systems [7], including confidentiality and integrity [40], authentication [41], and even trust [42] for decentralized systems [43, 44, 45].

6. Conclusions and future work

In this paper, we initiated a taxonomy of hyperproperties allowing to address both safety and security issues in RFID-based infrastructures. As future work, we plan to extend our classification with information flow policies relevant for the IoT context, also in view of applications to the analysis and verification of systems and protocols based on wireless sensor networks and mobile ad-hoc networks. We then intend to proceed to the algorithmic verification of the hyperproperties obtained in existing model checkers[1]. Finally, it would be interesting to investigate counterexamples in case of policy violations [47].

Acknowledgments

This work has been funded by the European Union - NextGenerationEU within the framework of PNRR Mission 4 - Component 2 - Investment 1.1 under the Italian Ministry of University and Research (MUR) programme "PRIN 2022" - grant number 2022598LMZ - AsCoT-SCE - CUP: H53D23003430006.

References

[1] K. Finkenzeller, RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and near-Field Communication, 3rd ed., Wiley, Hoboken, 2010.

[1]For instance, MCHyper [46], available at https://finkbeiner.groups.cispa.de/tools/mchyper/.

[2] N. Prabhu, Design and Construction of an RFID-enabled Infrastructure. The Next Avatar of the Internet, CRC Press, Boca Raton, 2013.
[3] A. Juels, RFID security and privacy: a research survey, IEEE Journal on Selected Areas in Communications 24 (2006) 381–394. doi:10.1109/JSAC.2005.861395.
[4] A. Khattab, Z. Jeddi, E. Amini, M. Bayoumi, RFID Security. A Lightweight Paradigm, Springer, Cham, 2016. doi:10.1007/978-3-319-47545-5.
[5] D. Bagay, Information security of RFID tags, Procedia Computer Science 169 (2020) 183–186. doi:https://doi.org/10.1016/j.procs.2020.02.133.
[6] M. R. Clarkson, F. B. Schneider, Hyperproperties, Journal of Computer Security 18 (2010) 1157–1210. doi:10.3233/JCS-2009-0393.
[7] T. v. Deursen, S. Radomirović, Security of RFID Protocols – A Case Study, Electronic Notes in Theoretical Computer Science 244 (2009) 41–52. doi:10.1016/j.entcs.2009.07.037.
[8] R. Focardi, F. L. Luccio, Secure Recharge of Disposable RFID Tickets, in: G. Barthe, A. Datta, S. Etalle (Eds.), Formal Aspects of Security and Trust. FAST 2011, volume 7140 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2011, pp. 85–99. doi:10.1007/978-3-642-29420-4_6.
[9] G. Barthe, C. Fournet, B. Grégoire, P.-Y. Strub, N. Swamy, S. Zanella-Béguelin, Probabilistic relational verification for cryptographic implementations, in: Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, ACM, New York, NY, USA, 2014, p. 193–205. doi:10.1145/2535838.2535847.
[10] D. L. Li, A. Tiu, Combining ProVerif and Automated Theorem Provers for Security Protocol Verification, in: P. Fontaine (Ed.), Automated Deduction – CADE 27. CADE 2019, volume 11716 of *Lecture Notes in Computer Science*, Springer, Cham, 2019, pp. 354–365. doi:10.1007/978-3-030-29436-6_21.
[11] Q. L. Meunier, A. R. Taleb, VerifMSI: Practical Verification of Hardware and Software Masking Schemes Implementations, Cryptology ePrint Archive, Paper 2023/732, 2023. URL: https://eprint.iacr.org/2023/732.
[12] P. Popovski, Tree-Based Anti-Collision Protocols for RFID Tags, in: M. Bolić, D. Simplot-Ryl, I. Stojmenović (Eds.), RFID Systems: Research Trends and Challenges, Wiley, Hoboken, 2010, pp. 203–228. doi:10.1002/9780470665251.ch8.
[13] J. McLean, A general theory of composition for a class of "possibilistic" properties, IEEE Transactions on Software Engineering 22 (1996) 53–67. doi:10.1109/32.481534.
[14] N. Coenen, B. Finkbeiner, C. Hahn, J. Hofmann, The Hierarchy of Hyperlogics, in: Proc. of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '19), Article No.: 39, IEEE, 2019, pp. 1–13. doi:10.1109/LICS.2019.8785713.
[15] K. Ali, A.-E. M. Taha, H. S. Hassanein, Medium Access Control in RFID, in: Y. Zhang, L. T. Yang, J. Chen (Eds.), RFID and Sensor Networks. Architectures, Protocols, Security and Integrations, CRC Press, Boca Raton, 2010, pp. 3–30.
[16] J. F. Hayes, An Adaptive Technique for Local Distribution, IEEE Transactions on Communications 26 (1978) 1178–1186. doi:10.1109/TCOM.1978.1094204.
[17] B. Tsybakov, V. Mikhailov, Free Synchronous Packet Access in a Broadcast Channel with Feedback, Problemy Peredachi Informatsii 14 (1978) 32–59. URL: https://www.mathnet.ru/eng/ppi1558.
[18] J. I. Capetanakis, Tree algorithms for packet broadcast channels, IEEE Transactions on Information Theory 25 (1979) 505–515. doi:10.1109/TIT.1979.1056093.
[19] L. Lamport, Proving the Correctness of Multiprocess Programs, IEEE Transactions on Software Engineering SE-3 (1977) 125–143. doi:10.1109/TSE.1977.229904.
[20] J. A. Goguen, J. Meseguer, Security policies and security models, in: Proc. of the 1982 IEEE Symposium on Security and Privacy, IEEE, 1982, pp. 11–20. doi:10.1109/SP.1982.10014.
[21] R. Focardi, R. Gorrieri, Classification of Security Properties (Part I: Information Flow), in: R. Focardi, R. Gorrieri (Eds.), Foundations of Security Analysis and Design. FOSAD 2000, volume 2171 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2001, pp. 331–396. doi:10.1007/3-540-45608-2_6.
[22] A. Pnueli, The temporal logic of programs, in: Proc. of the 18th Annual Symposium on Foundations

of Computer Science (SFCS 1977), 1977, pp. 46–57. doi:10.1109/SFCS.1977.32.
[23] J. A. W. Kamp, Tense Logic and the Theory of Linear Order. Ph.D. Thesis, UCLA, 1968.
[24] D. Gabbay, A. Pnueli, S. Shelah, J. Stavi, On the temporal analysis of fairness, in: Proc. of the 7th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '80, Association for Computing Machinery, New York, NY, USA, 1980, pp. 163—-173. doi:10.1145/567446.567462.
[25] I. Hodkinson, Expressive Completeness of Until and Since over Dedekind Complete Linear Time, in: A. Ponse, M. de Rijke, Y. Venema (Eds.), Modal Logic and Process Algebra. A Bisimulation Perspective, volume 53 of *CSLI Lecture Notes*, CSLI Publications, Stanford, 1995.
[26] A. Rabinovich, A Proof of Kamp's theorem, Logical Methods in Computer Science 10 (2014) 1–16. doi:10.2168/LMCS-10(1:14)2014.
[27] M. R. Clarkson, B. Finkbeiner, M. Koleini, K. K. Micinski, M. N. Rabe, C. Sánchez, Temporal Logics for Hyperproperties, in: M. Abadi, S. Kremer (Eds.), Principles of Security and Trust. POST 2014, volume 8414 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2014, pp. 265–284. doi:10.1007/978-3-642-54792-8_15.
[28] B. Finkbeiner, M. Zimmermann, The First-Order Logic of Hyperproperties, in: H. Vollmer, B. Vallée (Eds.), Proc. of the 34th Symposium on Theoretical Aspects of Computer Science (STACS 2017), volume 66 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2017, pp. 30:1–30:14. doi:10.4230/LIPIcs.STACS.2017.30.
[29] E. Bartocci, T. Ferrère, T. A. Henzinger, D. Nickovic, A. O. da Costa, Flavors of Sequential Information Flow, in: B. Finkbeiner, T. Wies (Eds.), Verification, Model Checking, and Abstract Interpretation. VMCAI 2022, volume 13182 of *Lecture Notes in Computer Science*, Springer, Cham, 2022, pp. 1–19. doi:10.1007/978-3-030-94583-1_1.
[30] B. Finkbeiner, C. Hahn, Deciding Hyperproperties, in: J. Desharnais, R. Jagadeesan (Eds.), Proc. of the 27th International Conference on Concurrency Theory (CONCUR 2016), volume 59 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2016, pp. 13:1–13:14. doi:10.4230/LIPIcs.CONCUR.2016.13.
[31] B. Bonakdarpour, B. Finkbeiner, The Complexity of Monitoring Hyperproperties, in: Proc. of the 2018 IEEE 31st Computer Security Foundations Symposium (CSF), IEEE, 2018, pp. 162–174. doi:10.1109/CSF.2018.00019.
[32] A. Rahmati, L. Zhong, M. Hiltunen, R. Jana, Reliability Techniques for RFID-Based Object Tracking Applications, in: Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07), 2007, pp. 113–118. doi:10.1109/DSN.2007.81.
[33] E. Börger, E. Grädel, Y. Gurevich, The Classical Decision Problem, Springer, Heidelberg, 1997.
[34] D. McCullough, Specifications for Multi-Level Security and a Hook-Up Property, in: Proc. of the 1987 IEEE Symposium on Security and Privacy, IEEE, 1987, pp. 161–161. doi:10.1109/SP.1987.10009.
[35] G. Boudol, I. Castellani, Noninterference for concurrent programs and thread systems, Theoretical Computer Science 281 (2002) 109–130. doi:https://doi.org/10.1016/S0304-3975(02)00010-5.
[36] A. Sabelfeld, A. C. Myers, Language-based information-flow security, IEEE Journal on Selected Areas in Communications 21 (2003) 5–19. doi:10.1109/JSAC.2002.806121.
[37] R. Giacobazzi, I. Mastroeni, Generalized Abstract Non-interference: Abstract Secure Information-Flow Analysis for Automata, in: V. Gorodetsky, I. Kotenko, V. Skormin (Eds.), Computer Network Security. MMM-ACNS 2005, volume 3685 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2005, pp. 221–234. doi:10.1007/11560326_17.
[38] S. Tini, Rule formats for compositional non-interference properties, The Journal of Logic and Algebraic Programming 60-61 (2004) 353–400. doi:10.1016/j.jlap.2004.03.003.
[39] A. Esposito, A. Aldini, M. Bernardo, Branching Bisimulation Semantics Enables Noninterference Analysis of Reversible Systems, in: M. Huisman, A. Ravara (Eds.), Formal Techniques for Distributed Objects, Components, and Systems. FORTE 2023, volume 13910 of *Lecture Notes in*

Computer Science, Springer, Cham, 2023, pp. 57–74. doi:10.1007/978-3-031-35355-0_5.

[40] H. Mantel, Information Flow and Noninterference, in: H. C. A. van Tilborg, S. Jajodia (Eds.), Encyclopedia of Cryptography and Security, Springer US, Boston, 2011, pp. 605–607. doi:10.1007/978-1-4419-5906-5_874.

[41] R. Focardi, R. Gorrieri, F. Martinelli, Message Authentication through Non Interference, in: T. Rus (Ed.), Algebraic Methodology and Software Technology. AMAST 2000, volume 1816 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2000, pp. 258–272. doi:10.1007/3-540-45499-3_20.

[42] F. Martinelli, M. Petrocchi, A Uniform Framework for Security and Trust Modeling and Analysis with Crypto-CCS, Electronic Notes in Theoretical Computer Science 186 (2007) 85–99. doi:10.1016/j.entcs.2007.03.024.

[43] R. Casadei, A. Aldini, M. Viroli, Combining Trust and Aggregate Computing, in: A. Cerone, M. Roveri (Eds.), Software Engineering and Formal Methods. SEFM 2017, volume 10729 of *Lecture Notes in Computer Science*, Springer, Cham, 2018, pp. 507–522. doi:10.1007/978-3-319-74781-1_34.

[44] A. Aldini, J.-M. Seigneur, C. Ballester Lafuente, X. Titi, J. Guislain, Design and validation of a trust-based opportunity-enabled risk management system, Information and Computer Security 25 (2017) 2–25. doi:10.1108/ICS-05-2016-0037.

[45] A. Aldini, S. M. B. Maranhão Moreno, J.-M. Seigneur, A Rule-Language Tailored for Financial Inclusion and KYC/AML Compliance, in: 2023 20th Annual International Conference on Privacy, Security and Trust (PST), 2023, pp. 1–10. doi:10.1109/PST58708.2023.10320148.

[46] B. Finkbeiner, M. N. Rabe, C. Sánchez, Algorithms for Model Checking HyperLTL and HyperCTL*, in: D. Kroening, C. S. Păsăreanu (Eds.), Computer Aided Verification. CAV 2015, volume 9206 of *Lecture Notes in Computer Science*, Springer, Cham, 2015, pp. 30–48. doi:10.1007/978-3-319-21690-4_3.

[47] N. Coenen, R. Dachselt, B. Finkbeiner, H. Frenkel, C. Hahn, T. Horak, N. Metzger, J. Siber, Explaining Hyperproperty Violations, in: S. Shoham, Y. Vizel (Eds.), Computer Aided Verification. CAV 2022, volume 13371 of *Lecture Notes in C0mputer Science*, Springer, Cham, 2022, pp. 407–429. doi:10.1007/978-3-031-13185-1_20.