Information Flow Control for the Internet of Things

Gildas Kouko, Josée Desharnais and Nadia Tawbi
Université Laval, 2325 rue de l'Université, Québec, QC, G1V 0A6, Canada
{*gildas.kouko.1, josee.desharnais.1, nadia.tawbi.1*}*@ulaval.ca*

Keywords: Internet of Things, Information Flow, Reachability Property, Safety Property, Model Checking.

Abstract: The Internet of Things (IoT) refers to devices and applications that interact and connect the physical and digital worlds. Unfortunately, their interactions often lead to information leaks and safety issues. Controlling their autonomous behavior related to events and actions in their environment is therefore important. This is the key to uncovering conflicts between user-defined expectations. To control these conflicts, we propose to verify IoT information-flow according to the principles of model checking. We propose a model based on an abstraction of the information flow induced by device and application operations and interactions in an IoT network. More precisely, the model is independent of any functional and technical heterogeneity. This abstraction is the result of an information flow analysis carried out, a priori, for all the involved devices, as well as for all the applications controlling them. A transition system is constructed from these abstractions enabling us to transform the information flow control into reachability and safety properties verification. We express these properties using a modal logic inspired by Timed Computation Tree Logic (TCTL) (Baier and Katoen, 2008). We illustrate our approach with an example and adapt the language and the model to an existing model checker.

1 INTRODUCTION

The Internet of Things (IoT) connects the physical and digital worlds. With the flourishing of IoT, social networks, the advent of cloud computing and the spread of mobile information access technologies, information is becoming increasingly available. It can be found everywhere instantaneously with digital networks of applications and devices that interact and communicate with each other. However, IoT networks may cause information leakage or illegitimate information flows i.e., the exposure of any type of confidential, sensitive or protected information (example: audio, image, integer, text, etc.) to an unauthorized user, applications or devices (Park et al., 2019; Sha et al., 2018). Also, the interacting IoT applications and devices could be developed or maintained using the same IoT platform or different IoT platforms such as Samsung SmartThings, Apple HomeKit, etc. (Abuserrieh and Alalfi, 2024). But when they run interactively in one environment, it may cause serious safety issues. In this extremely complex context of multitudes of devices and applications connected, information flow control has become an constant concern, given the massive adoption of IoT and the convenience it brings.

There are several examples of information leakage from devices or applications in the literature (Cardoza, 2016; Zetter, 2014; Neagle, 2015). But the prevalence of vulnerabilities in cloud-hosted services is particularly problematic for public devices (Mafamane et al., 2021). Users sometimes configure or connect devices and applications to services inappropriately. They lack the knowledge to perceive the subtle interactions of devices and applications and have a poor understanding of their functionality, partly because of their imprecise description. Moreover, users are increasingly connecting devices and applications that become visible to each other. This does not provide them with a comprehensive understanding of how systems work, especially when it comes to installing and configuring applications from different companies.

Users use interfaces to express automation rules, hoping to obtain an information flow that reflects their ideas. If these devices and applications interact and produce a lot of information, controlling the flow of their exchanges is not a simple matter. The more connected devices there are, the greater the risk of interference between automation rules, leading to unexpected behavior such as information leaks or safety issues (Reddit, 2018). They are also heterogeneous and use different protocols or information representation schemes with varying semantics (Sobin, 2020).

Kouko, G., Desharnais, J. and Tawbi, N.
Information Flow Control for the Internet of Things.
DOI: 10.5220/0013297800003944

In *Proceedings of the 10th International Conference on Internet of Things, Big Data and Security (IoTBDS 2025)*, pages 105-116
ISBN: 978-989-758-750-4; ISSN: 2184-4976

Heterogeneity in IoT systems can refer to the diversity of IoT devices, networks, applications and platforms.

Information leakage poses a confidentiality issue. Solutions based on access control and cryptography are available to protect the confidentiality of information transmissions and limit unauthorized information disclosure (Ngo and Nguyen, 2019). These approaches prevent the reading or modification of confidential information by unauthorized users, and dually prevent untrusted flow of information to compromise the integrity of trusted information. However, these techniques do not control the interference of automation rules in an IoT network. They do not control also how the information, once decrypted or known, is propagated at the risk of implying an information leak or violating predefined rules. These limitations together with verifying the safety of an IoT system are the main issues addressed in our work, which we clarify in Section 2.

The general purpose of our work is to enforce adequate information flow and predefined safety properties in IoT systems. We want to provide tools to show that a system actually applies the rules predefined by its users, regardless of the underlying platform. This requires to provide a rigorous analysis of the integrated behavior of devices and applications, rather than each in isolation, when connected in a given configuration. In this way, users could systemically validate their configuration intentions through the communication channels they themselves establish by properly interconnecting the devices and applications in their IoT system.

In this paper, we propose a new approach to control IoT information flow via formal verification of safety and reachability properties. Reachability properties express the fact that a desirable configuration is always reachable from an initial configuration and safety properties state that something bad should never happen i.e., the system is not dangerous for itself or its environment as long as the assumptions on which the system is built are fulfilled. Our approach consists of three steps, the first two being the following contributions:

- We use an abstraction of devices and applications behaviors to represent interactions in an IoT network, as a model that a user can draw. The model reflects the relations between communication ports and the information flow from one application or device to another. The abstraction is the result of a priori analysis of the information flows for all the devices analyzed, all the applications that control them, as well as the representation of the interconnections in the configuration adopted by the user. We specify this analysis's result in Section 4.2. The model built from these abstractions allows to transform the information flow control problem into a problem of reachability. Furthermore, this model naturally allows the verification of safety properties. Interconnecting devices and applications does not guarantee that a user's expectations are met. These expectations must also be verified against the abstract model.
- In order to use a formal approach to verify the safety and reachability properties based on our model, we express them in a new formal language, inspired by TCTL (Timed Computation Tree Logic) (Baier and Katoen, 2008), that is intuitive and easy for everyone to use. The originality of this language lies in the fact that it can express both high-level and specific properties with devices or applications communication attributes.

The final step is to automatically verify that the model satisfies the reachability or safety properties using a model checker. This requires an acceptable adaptation of our model and language. We present these adaptations in section 4.5.

Section 2 returns to the issues to be solved in more details. We present related work in Section 3, Our approach of controlling information flow is presented in Section 4. We illustrate our approach with an example in Section 4.6. Further discussion and observations to extend our work are presented in Section 5. Finallay, we conclude the paper in Section 6.

2 PROBLEM STATEMENT

In an IoT network, we want to guarantee confidentiality in information exchanges and the safety of an IoT system. But this guarantee must be based on user-defined automation rules, general rules, laws, or any user defined or context induced policy. To better understand the issue at hand, let us note that a device or application can acquire and generate information and act on its environment. The device's resources therefore include various interfaces enabling it to exchange information over various wired or wireless networks through input and output ports attached to communication protocols (Yu et al., 2020; Mainuddin et al., 2021).

Even if IoT applications perform securely and safely when run in isolation, they may cause dependability issues when they run interactively in one environment. The interacting IoT applications can be developed or maintained using the same IoT platform or different IoT platforms. Referring back to the example of contradicting rules in IoT system, *when Smoke*

is detected, then activate the sprinkler. When moisture is detected, switch off the water valve. If both smoke detectors and smart valve devices work interactively in the same environment, we have to consider the shared environment properties. The action of one device triggers an event associated with another device that is set up in the same environment. The resulting contradiction may cause serious safety hazards if such behavior is not prohibited or fixed. Foremost, one aspect of our work is to guarantee the reliable operation of the devices and applications involved in an IoT network.

As we have mentioned, traditional methods such as access control, firewalls and cryptography impose limits on access to information but provide no control over the confidentiality or propagation of information once it has been decrypted or made accessible to a device, application or user. In an IoT network, devices and applications can generate new information for transmission based on information received on different communication ports. So, they can influence each other and so do communication ports. In this way, if we want to check, for instance, whether an image passes from a device D_1 to a device D_2, in the case where an application or device D_3 lies between the two, it is important to know whether or not this data is relayed through D_3. The influence relations between the input and output communication ports of D_1, D_2 and D_3 are therefore important characteristics to be considered when studying information flows of an IoT system. In this work, we aim at exploiting these influence relations to control confidentiality in information exchanges in an IoT system. We designate input and output entries by ports whether it is related to a hardware device or to an application. We assume that a priori analyses have been performed in order to determine influences relating input ports to output ports for each device or application.

Devices and applications communicate together to meet a usage scenario based on desired safety and reachability properties of an IoT system. By a reachability property, we mean that an information is allowed or not to reach some destination (a device or an application or a user). To illustrate our information flow control approach, we will consider the following use-case for configuring a smart system in a building.

Example 2.1. *Let us consider a building equipped with a smart system featuring an application App, three cameras* C1*,* C2 *and* C3 *each equipped with motion detectors, two televisions* Tv1 *and* Tv2 *and a sound system* S*. The application records images from the cameras, and then processes and broadcasts them to the televisions. In the event of an incident notification, the application sends a signal to all devices and to the application. The sound system uses the speakers* Sp1 *and* Sp2 *and the television* Tv1 *to broadcast music. To complete the smart system description, Table 1 presents some of the non-exhaustive amenities expected in the building and Table 2 lists a few desired safety and reachability properties.*

Table 1: Some amenities of a smart building.

	Amenities
1.	A dance hall equipped with Speaker Sp1 and an emergency exit door.
2.	A childcare room equipped with Speaker Sp2 and Camera C3.
3.	An entrance hall accessible only through two doors, each equipped with a camera capable of relaying signals. C1 is for the first door and C2 is for the second one.
4.	A control room equipped with Televisions Tv1 and Tv2 to monitor entrances and exits in the hall and children's motions in the childcare room.
5.	A room equipped with Sound system S linked to the two speakers and to television Tv1.

Table 2: Some configuration properties.

Names	Properties
P.1	If the childcare room camera, C3, transmits an image, it appears on the control childcare television, Tv2.
P.2	If the application sends an off signal, the emergency door closes.
P.3	Opening the emergency door should shut down the sound system S.
P.4	the images from the cameras on the entrance hall doors never reach the control childcare television Tv2.
P.5	If a camera detects a motion, the application receives an image.
P.6	The music played on the surveillance television of the lobby is always the same as that played on the speaker in the dance hall.

In the recent years, various researchers have proposed a variety of architecture reference models for building the IoT systems. The models were explained and comparative analysis of these models was achieved (Frappier et al., 2010; Stevens et al., 2020; Arslan et al., 2023). The focus of these studies is to build a model of the IoT from the dimension of system function and composition, and get the nature of the IoT system through theoretical analysis. None of these studies provides formal modeling and verification methods for the integration of devices and ap-

plications taking into account, at the same time, the global communication and their internal communication. Our approach provides a good and easy reference for the design and model validation of IoT systems using transition systems. This approach allows us to achieve better modularity when we add or retrieve new applications or smart devices. Since we have only to represent how a new element is interconnected to the network and to provide an abstraction of its inner communications. Devices and applications are represented as information vehicles. They receive, generate and transmit information, which may depend on the information they receive. It considers fully information flow in the interactivity with physical world despite devices and applications heterogeneity. Our approach verification performance depends directly on the model checker used. In the related work 3, which cuts across several areas such as privacy and safety, we describe research in the literature.

3 RELATED WORK

Reliable information flow in an IoT network does not always depend on devices, applications or platforms. Users are sometimes a source of risk, as they consume output information from systems, but can also provide input information. To this end, Nunes et al. (Nunes et al., 2015) emphasize that for a system to better respond to human needs, its modeling and analysis must take into account human intentions, psychological states, emotions and actions deduced from sensory data. This is a particularly difficult challenge, given the complexity of the psychological and behavioral aspects of human beings. In addition to the aspect of modeling human behavior, managing the unpredictability and occasional unreliability of human information in an IoT environment makes it all the more complex to guarantee an information flow without information leakage. Lee and Truong (Lee and Truong, 2016) propose a management model of three confidence measures to reason about humans and identify potential causes of human failures in systems analysis. However, outstanding questions relate to how to find and use a standard human model to guarantee the reliability of information flows, given the wide human variety.

In addition to the difficulties associated with human behavior, the heterogeneity of IoT has resulted in fragmented systems as shown by Zhang (Zhang, 2017) and ADT's website (ADT, 2022). Unfortunately, this leads to owner solutions that suffer from a lack of testing, leaving users exposed to potential attacks from outdated software (Radovici et al., 2020). Bannour and Lapitre (Bannour and Lapitre, 2020) propose a model based on automata to promote rapid propagation of new firmware versions and thus minimize the periods during which devices are obsolete. Abuserrieh and Alalfi (Abuserrieh and Alalfi, 2024) and Gani et al. (Gani et al., 2015) focusing on the needs of users such as devices or applications use formal methods for testing in IoT environments. Unfortunately, Esquiagola et al. (Esquiagola et al., 2017) justify the inadequacy of current software testing methods to observe and control the inappropriate exposure of confidential information resulting from the actual interaction of devices and applications with the physical world. This is obvious in reviews outlining IoT issues related to information flow confidentiality, safety and security (Alhirabi et al., 2021; Li and Xu, 2017; Pradeep and Kant, 2022). These reviews are often focused on device or protocol faults, malicious applications, platform problems, and so on (Chen et al., 2018; Ang and Seng, 2019). However, there is few work dealing with IoT interaction control.

Clements et al. (Clements et al., 2017) propose a novel technique called privilege overlaying to apply protections against code injection, control-flow hijack, and data corruption attacks in a system. Chi et al. (Chi et al., 2020) employ symbolic execution to extract automation rules from IoT applications and utilize the analysis of abstract syntax tree to identify cross-app interference threats. They introduce a risk ranking technique that evaluates the severity of detected threats based on the impact of rule execution, functionality, and the criticality of device safety. Nguyen et al. (Nguyen et al., 2018) focus on analyzing IoT application interactions through the construction of dependency graphs. Wang et al. (Wang et al., 2018) use provenance-based tracing to generate a data provenance graph to perform the analysis in order to provide a complete history of IoT system interactions, which may guide the analysis to a potential malicious behavior. Abdelouadoud and Logrippo (Abdelouadoud and Logrippo, 2024) offer an approach to information access control that assigns roles to users and devices based on authorization levels.

In addition, developers struggle to decipher the semantics of the communication protocols of each device or application. Despite control structures in IoT application are limited to IFTTT (If This Then That) or triger-action programming (Abuserrieh and Alalfi, 2024), it is therefore not easy to find a standard method of extracting from the code source of any device or application the permissions linked to events and actions, in order to detect information leaks, and control interactions to avoid properties violations. Ce-

lik et al. (Celik et al., 2018) verify the compliance of a transition system extracted by static analysis to a set of user-predefined safety and confidentiality properties. In another paper, Celik et al. (Celik et al., 2019) extend their work to dynamic analysis in order to detect properties violations in an individual IoT application or a group of IoT application interacting in one environment. Users have the option either to allow the violations to be executed or to allow to block them. To identify properties violations via model checking, McCall et al. (McCall et al., 2023) model the rules of IoT applications as a labeled transition system and define the desired behavior by representing properties in the computational tree logic, where properties are typically specified as formulas in temporal logic (Baier and Katoen, 2008).

Given the aforementioned challenges, the above works are insufficient to theoretically analyze IoT systems and deal with information flow control as we do in our approach to verifying user-defined expectations.To the best of our knowledge, our modeling strategy is absent from the literature and therefore not comparable with any other.

4 INFORMATION FLOW CONTROL

An IoT network is made up of IoT devices and applications. Information therefore travels, through communication ports, both within and between network components. This information flow becomes illicit when it can cause information leakage i.e., a sensitive information reaches an unauthorized destination. Non-interference is the basic confidentiality property for information flow (Goguen and Meseguer, 1982; Sabelfeld and Myers, 2003). We want to enforce non-interference on the information flow of the IoT network. We propose an abstract representation of the connection of devices and application with a transition system. This allows to transform the non-interference problem to a reachability problem.

4.1 Basic IoT System Modeling

When connecting the devices and applications of a network, one gets a model that can be represented naturally with a transition system. Devices and applications are represented as vertices, and arrows symbolize oriented communication channels (links between two ports) specifying the information flow between vertices. Thus the arrows represent the physical connections, or wireless connections, that are established by the users.

Figure 1 is the transition system of the smart system of Example 2.1. The labels on the arrows are primarily intended to make the model easier to read. Labels of the same type and color identify similar information. The state's colors represent the rooms where the devices are. The sound system S is in the music control room (cyan), Sp1 is the speaker in the parents' room (orange), Sp2 is the speaker in the child-care room (magenta); Ci is the camera on hall door i, with $i = 1,2$, and C3 is the one in the children care room. Tv1 broadcasts the image of C1 and C2, in the security room (red) and Tv2 broadcasts the image of C3. Looking at Figure 1, we see that cameras C1 and C2 seem to influence the application directly and the television Tv1 indirectly. This corresponds to the description in Example 2.1. This model can be formalized with a system of transitions as shown in the following definition.

Definition 4.1. *A basic model of an IoT system is a tuple* $T = (S, O, I, L_1, \delta)$ *where:*

- S *is the set of devices and applications of the system,*
- O *is a set of output ports of format t.id!, where the letter t is the port type (examples: audio, image, signal, etc.), id is its unique identifier and the symbol ! indicates that it is an output port,*
- I *is a set of input ports of format t.id?, where t is the port type (examples: audio, image, signal, etc.), id is its unique identifier and the symbol ? indicates that it is an input port,*
- $L_1 : S \to 2^X$ *where* $X = I \cup O$ *is a labeling function that associates a finite set of ports with a vertex,*
- $\delta : S \times O \to S \times I$ *is a transition function. We write*

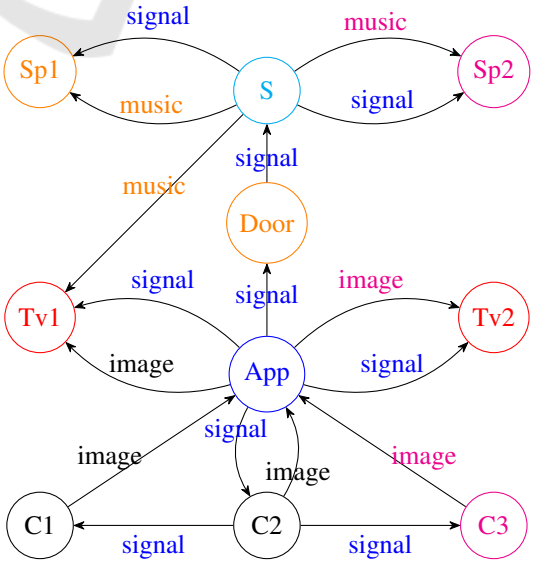

Figure 1: Basic model for the smart system of Example 2.1.

$s \xrightarrow{o,i} q$ *rather than* $\delta(s, o) = (q, i)$ *where* $o \in L_1(s)$ *and* $i \in L_1(q)$.

As information travels from one device or application to another, vertices and ports influence each other. We call this a basic model, since it does not reflect the influences between communication ports *inside* an application or a device. For instance, if an output port of a device is influenced only by a subset of the input ports, the basic model is unable to show this detail. In our example, one can think that the application transmits images from cameras C1 and C2 in the lobby to the television Tv2. This would break a configuration rule, even though this is not the case in the Example 2.1 description. In fact, such an error could arise from the user's incorrect connection of devices and the application. They do not necessarily know if the information input to the application influences the information output to Tv2. It is thus important to give them a more precise tool for validating their intentions expressed via the configuration of this system and the connection they actually establish.

4.2 Influence Matrix

As we argued, the basic modeling presented above is not sufficient, because the abstraction of internal devices and applications communication is too pessimistic. Thus we assume we are given an *influence matrix* for each device and application, that represents the possible flow of information from every input port to every output port within that device or application. This abstraction of information flow allows us to transform the non-interference problem to a reachability problem. For instance, a positive entry in the influence matrix between input port A to output port B means that *there may be* an information flow from port A to port B. If no public destination is reachable from private ones then the non-interference property is satisfied.

We assume that the influence matrix is given. It can be established using standard information flow analysis mechanisms. These can be divided into two categories: mechanisms based on a static analysis of the application, and mechanisms based on a dynamic approach (Damodaran et al., 2017). Some hybrid mechanisms also exist, usually using the results of static analysis when examining the code (Bedford et al., 2017). These mechanisms could be used upstream of our work. The following example illustrates the main application matrix of the smart building Example 2.1.

Example 4.1. *Let us consider Example 2.1 again. Table 3 is the influence matrix of the application. It tells us that the* App_a *port influences the* App_d *port, because the information received by the* App_a *port is sent, perhaps after processing, to the* App_d *port. The information sent by* App_d *port is therefore dependent on the information received by* App_a *port. The absence of the ✓ symbol in a table cell indicates that the ports concerned are independent of each other. It could also be that the object or application generates output information that is not influenced by the input port (example: a timer that sends a signal at a certain frequency to its output port). However, in the network, perhaps the ports influence each other through another loop.*

Such an influence matrix of smart objects could be supplied by IoT device manufacturers, downloaded from their sites, or deduced from the manufacturer specification. The influence matrix of an application could also be provided by its provider, or obtained by performing information flow analysis on the code of this application. They enable a user unaware of the interweaving of information within devices to perform specific diagnostics or validate their expectations and have counter-examples for those unsatisfied.

Table 3: Application influence matrix.

		Output port		
		App_d	App_e	App_f
Input port	App_a	✓		
	App_b	✓		
	App_c		✓	

4.3 The Extended IoT System Model

Combining the basic model, which represents how the objects and applications are linked, and the influence matrix, we define the extended model. Unlike the basic model, it must be automatically generated for the user by a software tool that integrates our concept and formalizes it as shown in the following definition.

Definition 4.2. *Let* $T = (S, O, I, L_1, \delta)$ *be a basic model. Its* extended model *is* $T' = (P, M, L_2, \delta', T)$ *where:*

- $P \subseteq O \cup I$ *is the set of communication port.*
- M *is a finite set of influence matrices such that* $\|M\| = \|S\|$ *and* $\forall m \in M, m_{i,o} \in \{true, false\}$ *represents the possibility of a flow from port* $i \in I$ *to port* $o \in O$.
- $L_2 : S \to M$ *is a labeling function that associates an influence matrix to each device and application of S,*
- $\delta' : P \to P$ *is a transition relation such that if* $\delta'(p) = p'$ *then* $\exists m \in M.m_{p,p'} = true$ *(in-*

ternal channel inside an element of S) or $\exists s,q \in S.\ s \xrightarrow{p,p'} q$ *(outer channel between two elements of S).*

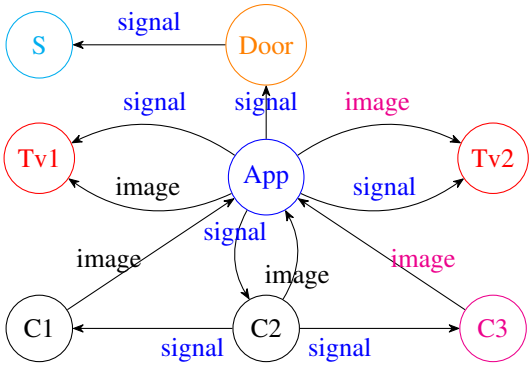

Figure 2: Extract of the model from the Figure 1.

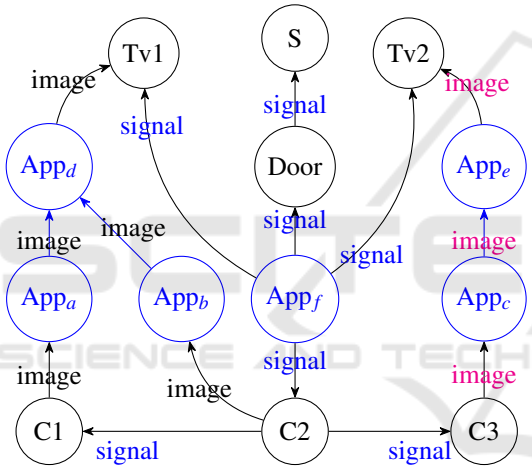

Figure 3: Extended information flow model.

It is with the extended model that we will use the model checker to validate more precisely that its basic model satisfies reachability or safety properties. We illustrate the extended model in Figure 3. It results from the model in Figure 2 extracted from the model in Figure 1 of the building Example 2.1. In the Figure 2, we only substitute the application with their communication ports, so as not to weigh down the extended model. The extended model in Figure 3 shows a transition system representing the connection of the application to the cameras, televisions and emergency door, as well as the connection of the door to the sound system. It focuses attention on the application (blue color), illustrating its substitution by its ports. The labels on the arrows serve only to help the reader understand the model, and those of the same color identify similar information. The application represented by the blue vertex in Figure 1 is split into six vertices, also in blue, representing its input and output ports, even if they are not physical ports. The cameras C1, C2 and C3 are connected to the ports App_a, App_b and App_c respectively. The television Tv1 is connected to the ports App_d and App_f and the television Tv2 is connected to the ports App_e and App_f. The sound system is connected to the door and the door is connected to the port App_f. The labels on the arrows are only there to help the reader understand the model.

According to the application influence matrix described in Table 3, the information received at App_a port influences that leaving at App_d port. That is why there is an arrow from App_a to App_d. So, in the model shown in Figure 3, we have an integration of the devices connection system with the influence matrix, enabling us to analyze how information travels through the network under study. The camera C1 therefore only influences the television Tv1 via the path $\text{App}_a \rightarrow \text{App}_d$ in the application and not both televisions as we might have thought if we had only looked at Figure 1.

4.4 Expressing Safety and Reachability Properties

In the context of system analysis, we are particularly interested in properties of safety and reachability. These properties are expressed in terms of sequences of states, or paths. In our setting, we want to focus on the paths that, when entering an input port, go out through a port that is indeed influenced by the input port, as expressed by the influence matrix. We thus formalize the notion of influence path in the following definition, to avoid any ambiguity in their evaluation.

Definition 4.3. *Let* $T' = (P, M, L_2, \delta', T)$ *be an extended model of T. An influence path in* T' *is a finite alternating sequence of devices (or applications) and communication ports* $\pi = s_0 p_0 p'_0 \ldots s_1 p_1 p'_1 s_3 p_3 p'_3$ *where* $\forall i \in \mathbb{N}$*:*

- $p_i \in L_1(s_i)$ *and* $p'_i \in L_1(s_{i+1})$
- $s_0 \xrightarrow{p_0,p'_0} s_1 \xrightarrow{p_1,p'_1} s_3 \xrightarrow{p_3,p'_3} \ldots$
- *if* $i \geq 1$ *and* $L_2(s_i) = m$ *then* $p'_{i-1} \in L_1(s_i)$ *and* $m_{p'_{i-1},p_i} = false$.

Ports abstraction in π *gives the influence path in T and we denote by :*

- $\pi(i)$ *the ith vertex of an influence path* $\pi\ \forall i \in \mathbb{N}$.
- *Path(s) the set of influence paths starting from* $s \in S$.

One could imagine describing properties in a natural language. Unfortunately, natural languages are too ambiguous for the needs of computerization, and not concise enough. Logics are formal specification languages. They have a precise mathematical definition to enable automatic verification of a system. One obvious challenge is to completely specify the behavior of a system, which is impossible to do automatically, especially given the multitude of devices and applications in an IoT network. To achieve this, we rely on families of properties. We propose a logic for expressing reachability and safety properties on basic and integrated models, which we call IFCTL (Information Flow Computation Tree Logic).

The aim of IFCTL is to scrutinize the information flow and the triggering of actions or events through influence paths one device or application to another. On the one hand, we want to express properties, intuitively and simply, at the level of a user unaware of the names of the ports used to send and receive information from devices or applications. On the other hand, the language allows properties to be stated by specifying port names, enabling the informed user to perform diagnostics on their model or to validate a counter-example given by a model checker when a property not being satisfied in the integrated model. IFCTL distinguishes between two classes of formulas: device or application formulas and system formulas. A device formula is interpreted on a device or application, as it provides a description of the communication ports. Information can be a signal, an action to be executed, an event, etc. It therefore has a type. A path formula is interpreted on an influence path. It is used to check the suitability of interactions between devices and applications. The following definition describes the syntax and semantics of IFCTL.

Definition 4.4. *Let $T' = (P, M, L_2, \delta', T)$ be an extended model of $T = (S, O, I, L_1, \delta)$. The syntax of IFCTL on $O \cup I$ is defined by:*

$$\psi ::= \varphi \mid \forall[\varphi : \varphi] \mid \forall[\varphi \,\square\, \varphi] \mid \exists[\varphi : \varphi] \mid \psi \wedge \psi \mid \neg\, \psi$$
$$\varphi ::= p \mid \mathit{true}$$

where φ is a device or application formula with $p \in O \cup I$ and $\varphi : \varphi$ is a path formula. The quantifier $\forall$ is read "for any influence path" and the quantifier $\exists$ is read "there is an influence path". Formula semantics are defined on the basis of the satisfaction relation $\models$. Let $s \in S$, then:

- $s \models p$ *iff* $p \in L_1(s)$
- $s \models \psi_1 \wedge \psi_2$ *iff* $s \models \psi_1$ *and* $s \models \psi_2$
- $s \models \neg\psi$ *iff* $\neg s \models \psi$
- $s \models \exists[\varphi_1 : \varphi_2]$ *iff* $\exists \pi \in Path(s).\pi \models \varphi_1 : \varphi_2$
- $s \models \forall[\varphi_1 : \varphi_2]$ *iff*
  $\forall \pi \in Path(s) \wedge \pi(0) \models \varphi_1.\pi \models \varphi_1 : \varphi_2$
- $s \models \forall[\varphi_1 \,\square\, \varphi_2]$ *iff*
  $\forall \pi \in Path(s) \wedge \pi(0) \models \varphi_1.\pi \models \varphi_1 \,\square\, \varphi_2$

For an influence path π then:

- $\pi \models \varphi_1 : \varphi_2$ *iff* $\pi(0) \models \varphi_1$ *and* $\exists i > 0.\pi(i) \models \varphi_2$
- $\pi \models \varphi_1 \square\, \varphi_2$ *iff* $\pi(0) \models \varphi_1$ *and* $\forall i \geq 0.\pi(i) \models \varphi_2$

In Definition 4.4, $\forall[\varphi_1 : \varphi_2]$ and $\exists[\varphi_1 : \varphi_2]$ express reachability properties. Table 4 illustrates the IFCTL formalization of some reachability and safety properties applicable to Example 2.1. The reachability properties P.1, P.2 and the safety property P.4 extracted from Table 2 are of high-level order. The reachability property P3 is more detailed, as it specifies communication ports. The absence of a port name in an IFCTL property means that any port can be used, which simplifies the formalization of the property. For example, $\text{Door}_{Close?}$ designates an input port of type *Close* and any name. The formalization of path properties identifies at least one initial port and one influenced port. In the property $\forall[\text{C1}_{Image.B!}{:}\text{Tv1}_{Image.B?}]$, $\text{C1}_{Image.B!}$ is the initial port and $\text{Tv1}_{Image.b?}$ is the sought-after destination. The mathematical quantifier $\forall$ indicates that we are interested of information flow paths from port $\text{C1}_{Image.b!}$ to port $\text{Tv1}_{Image.b?}$. IFCTL logic is in fact a subset of CTL (Computation Tree Logic) (Baier and Katoen, 2008), if one interprets ports as state labels.

To verify these properties on a model, we plan to use the model checker Uppaal (Uppsala and Aalborg, 2020). Developed jointly by the Uppsala and Aalborg Universities, Uppaal is a real-time systems model checker that has proven its effectiveness in several studies (Gerking et al., 2018; Chen et al., 2020). So we give a translation of the logic formulas into TCTL, a timed version of CTL that is accepted by Uppaal. Our plan is that this translation and the model checking will be done automatically, so that it appears transparent to the user.

Table 4: Some formal reachability and safety properties.

P.1	$\exists[\text{C3}_{Image!}{:}\text{Tv2}_{Image?}]$: if the childcare room camera C3 transmits an image, it appears on the control childcare television Tv2.
P.2	$\exists[\text{App}_{Signal.off!}{:}\text{Door}_{Close?}]$: if the application sends an off signal, the emergency door closes.
P.3	$\forall[\text{C1}_{Image.b!}{:}\text{Tv1}_{Image.b?}]$: the images transmitted by the b port of camera C1 arrive on television Tv1 via its port b.
P.4	$\forall[\text{C1}_{Image!} \square \neg \text{Tv2}_{Image?}] \wedge \forall[\text{C2}_{Image!} \square \neg \text{Tv2}_{Image?}]$: the images from the cameras on the entrance hall doors never reach the control childcare television Tv2.

4.5 IFCTL Translation to TCTL

TCTL expressive power lies in the fact that it is based on both temporal and path quantization (Baier and Katoen, 2008). It can also be used to determine the duration between two states or events. The following definition presents its syntax and semantics.

Definition 4.5. *TCTL ϕ formulas are recursively defined as follows:*

$$\phi ::= p \mid true \mid \phi \longrightarrow \phi \mid \forall[\phi_1 \bigcup_{\sim c} \phi_2] \mid \exists[\phi_1 \bigcup_{\sim c} \phi_2] \mid \phi \wedge \phi \mid \neg \phi$$

where p is an atomic proposition, $c \in \mathbb{N}$ and $\sim$ denotes one of the binary relations $<, \leq, =, \geq$ or $>$.

Informally, $\exists[\phi_1 \bigcup_{<c} \phi_2]$ describes a property indicating that there is an execution path where ϕ_1 is true until ϕ_2 it becomes and for a maximum of c time units. $\forall[\phi_1 \bigcup_{<c} \phi_2]$ means that for any execution path ϕ_1 remains true until ϕ_2 becomes true, and for a maximum of c time units. The following abbreviations are accepted in TCTL:

- $\exists\Diamond_{\sim c}\phi \equiv \exists[true \bigcup_{\sim c} \phi]$
- $\forall\Diamond_{\sim c}\phi \equiv \forall[true \bigcup_{\sim c} \phi]$
- $\forall\Box_{\sim c}\phi \equiv \neg\exists\Diamond_{\sim c}\neg\phi$

Unrestricted temporal operators correspond to operators subscribed with ≥ 0. For example, $\exists\Diamond\phi$ corresponds to $\exists\Diamond_{\geq 0}\phi$. $\exists\Diamond\varphi_1$ and $\forall\Diamond\varphi_1$ are translated respectively into $E <> \varphi_1$ and $A <> \varphi_1$ in the model checker Uppaal.

IFCTL's property formalism is similar to that of TCTL. However, the properties of the form $\forall[\varphi_1 : \varphi_2]$ and $\exists[\varphi_1 : \varphi_2]$ can not be translated directly into $\forall[\varphi_1 \bigcup \varphi_2]$ and $\exists[\varphi_1 \bigcup \varphi_2]$ in TCTL. In an extended model, on the one hand, if φ_1 identifies a single state, it can not be true all along an influence path, as TCTL semantics would have it. On the other hand, if φ_1 or φ_2 is a formula identifying several states, it should be reformulated in such a way as to be able to check whether there is an influence path from any state inherent in φ_1 to the states falling under φ_2. This would mean exploding $\forall[\varphi_1 : \varphi_2]$ or $\exists[\varphi_1 : \varphi_2]$ into a conjunction of formulas. Taking these remarks into account, we propose the essential equivalence of the satisfaction relation between the IFTCL and TCTL logics in the following definition.

Definition 4.6. *Let $T = (S, O, I, L_1, \delta)$, $T' = (P, M, L_2, \delta', T)$ an extended model of $\top$, $s \in S$, $p \in P$, φ_1 and φ_2 IFCTL device formulas, ψ_1 and ψ_2 are any IFCTL formulas and fnct is a function such that $fnct(\varphi_1)$ returns the set of ports identified by φ_1. The equivalence relation $\equiv$ starting from a vertex $s \in S$ is defined as follows:*

- $s \models \exists[\varphi_1 : \varphi_2] \equiv \exists p \models \varphi_1.p \models \exists\Diamond\varphi_2$
- $s \models \forall[\varphi_1 : \varphi_2] \equiv \forall p \models \varphi_1.p \models \forall\Diamond\varphi_2$

where $p \models \varphi_1$ iff $p \in fnct(\varphi_1)$

To check $s \models \psi_1 \wedge \psi_2$, use one of the two equivalence forms above to check $s \models \psi_1$ and $s \models \psi_2$. Any vertex in TCTL satisfies the propriety true.

To illustrate the result of the *fnct* function, $fnct(\text{App}_{Image!}) = \{\text{App}_d\ ,\ \text{App}_e\}$ according to the application influence matrix in Table 3.

4.6 Case Study

We devote this section to illustrate our information flow control approach in IoT systems. We use an extract from the Example 2.1 whose corresponding basic model is given by Figure 2. To build the corresponding extended model described in Figure 4, we need the influence matrices of devices and applications involved in the basic model. They are given by the influence matrices in Tables 3, 5 and 6. In everyday life, thanks to their sensors, the camera's motion detectors capture data from their surroundings and sent signals. Camera software is programmed to use available network connections to initiate an action based on this data. In this case, this means sending the data to the application for processing. It should be remembered here that it is rare for an information transmission port to be unaffected by an information input port. But, to keep Figure 4 as light as possible, we have omitted the motion sensors in the cameras whose signals trigger images capture. As a result, we have drawn up the influence matrices showing only those whose at least an input ports influence an output port. The ports of cameras C1 and C2 and televisions Tv1 and Tv2 are therefore independent of each other in their influence matrices, which are not presented here. The door's influence matrix is trivial. It has only one input port and one output port that influence each other. The two properties expected of the smart system are P.1 and P.3, set out in Table4.

Table 5: The camera 2 influence matrix.

		Output port	
		$C2_b$	$C2_c$
Input port	$C2_a$		✓

Table 6: The sound system influence matrix.

		Output port		
		S_b	S_c	S_d
Input port	S_a			✓

In the basic model shown in Figure 2, the verification of the property P.1 consists in checking if

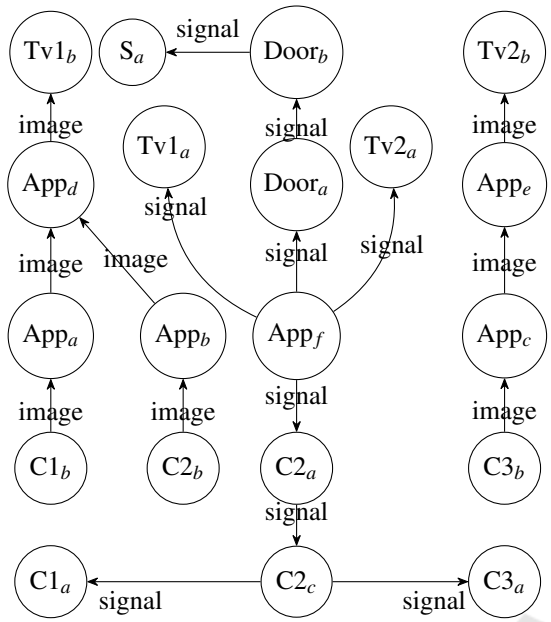

Figure 4: Extended model of the basic model in Figure 1.

C3 $\models E <>$ Tv2 and for property P.3 to check if $C_1 \models A <>$ Tv1. These properties are all true in the basic model, but at this stage of the verification, we are not yet sure. They must therefore be checked in the extended model shown in Figure 4 by checking if C3$_b \models E <>$ Tv2$_b$ (in IFCTL: C3$_b \models \exists\Diamond$ Tv2$_b$) for P.1 and if C1$_b \models A <>$ Tv2$_b$ (in IFCTL: C1$_b \models \forall\Diamond$ Tv2$_b$) for P.3. In the basic model, the property $E <>$ Tv1 is true for C1, C2 and C3. The property $E <>$ Tv1$_b$ is false for camera C3, as it can only send images to port Tv2$_b$, as shown in the extended model.

As we have developed an automatic verification approach, its performance is directly linked to the model checker. So we have to cope with the potential state-space explosion in our basic model extended model. As far as concern Uppaal, Larsen et al (Larsen et al., 1995) has shown via experimental results that is not only substantially faster than the other real-time verification tools but also able to handle much larger systems.

5 DISCUSSION

Some related work do not consider formalizing full automation of security and safety properties in IoT analysis. The focus is rather on analyzing the IoT system. Some tools use a text-based description of the properties and conduct manual conformance to IoT system analysis toward these properties. Some tools are based on transforming the IoT system into many representations, which affects the performance analysis. The lack of IoT security and safety standardization adds difficulties to this process as well. The originality of our work lies in the device's internal communications abstraction by their communication ports in order to analyze system behavior globally and identify inadequate information flows through model checking. This form of abstraction mitigates the impact of functional and technical complexities of IoT heterogeneity. Our concept of checking reachability and safety properties can be exported to several other IoT domains. Integrity enforcement is known to be dual to confidentiality enforcement (Schneider, 2000), instead of tracking if confidential information can reach public destinations, we track if corrupted information can reach trusted destinations. For example, we can check to which extent altered information is propagated in autonomous cars to a smart city IoT network. We assume that influence matrices are available from a priori analysis (Sabelfeld and Myers, 2003; Bedford et al., 2017) 4.2.

In this work, we have not integrated into our modeling the temporal aspect of device operation. Smart Devices collect or affect the physical environment depending on a certain frequency and period, so the service provided by the IoT system often has high dynamic and real-time requirements. An extension of our approach could build a complete model of an IoT system with temporal and state dimensions, like the use-case scenario proposed by Chen et al (Chen et al., 2020) in their paper. Such a new feature will certainly make IFCTL more expressive, as it will allow to express properties such as $\forall[\varphi :_{\sim c} \varphi]$, $\forall[\varphi \Box_{\sim c} \varphi]$ and $\exists[\varphi :_{\sim c} \varphi]$. The time unit c will be interpreted as in TCTL.

Furthermore, since the IoT environmental events are nondeterministic in nature (Choe and Lee, 2018), another extension could be to build a probabilistic extended model in which uncertainties and randomized behaviors are modeled by assigning probability values in the influence matrices (Kwiatkowska and Parker, 2012; Desharnais et al., 2002). Such modeling could be used in an IoT system composed of communicating robots equipped with multiple sensors that monitor each robot. The sensors record and communicate various parameters, such as time, pressure, temperature, etc. However, due to issues such as sensor malfunctions, network problems or storage errors, expected influences in the information flow could not always occur: hence the idea of assigning probabilistic values to the transitions in the targeted model. However, it is a challenge to guarantee completeness and maintain a high level of data quality (Klier et al., 2024). Uppsala 5.0 (Uppsala and Aalborg, 2020) can

be used for the future work proposed.

6 CONCLUSION

IoT systems are overgrowing, which results in more connected devices, more developed IoT applications and platforms. This growth introduces new challenges to IoT systems and their environments related to security and safety. The heterogeneity of IoT plateforms and networking technologies contribute largely to the complexity of these issues. Thus, there is an urgent need for appropriate tools and methods aiming at the design of IoT systems able to operate safely without leaking sensitive information.

In this paper, we propose to use model checking to control information flow in an IoT network. Our model abstracts from technical and functional heterogeneity and integrates influence relations between device and application communication ports. Our approach reduces the information leakage problem to a reachability and safety problem, in order to verify user expectations. We propose to express both general and specific properties in a new logic IFCTL inspired by TCTL, an existing language accepted by known model checkers. The model checker Uppaal accepts our basic and integrated models. We have illustrated our approach with a rich but well-chosen example to underline the problem we are solving. Our work does not rely on specific plateforms therefore it can be easily applied to several IoT application domains.

REFERENCES

Abdelouadoud, S. and Logrippo, L. (2024). Implementation of a partial order data security model for the internet of things (IoT) using software defined networking (SDN). *Journal of Cybersecurity and Privacy*, 4:468–493.

Abuserrieh, L. and Alalfi, M. (2024). A survey on verification of security and safety in IoT systems. *IEEE Access*.

ADT (2022). A comprehensive guide to smart home device compatibility, ADT's smart home security. https://www.adt.com/resources/smart-home-device-compatibility. Accessed: 2023-10-18.

Alhirabi, N., Rana, O., and Perera, C. (2021). Security and privacy requirements for the internet of things: A survey. *ACM Transactions on Internet of Things*, 2:1–37.

Ang, L.-M. and Seng, K. (2019). Application specific internet of things (asIoTs): Taxonomy, applications, use case and future directions. *IEEE Access*, PP:1–1.

Arslan, S., Ozkaya, M., and Kardas, G. (2023). Modeling languages for internet of things (iot) applications: A comparative analysis study. *Mathematics*, 11:1263.

Baier, C. and Katoen, J. (2008). *Principles of model checking*. MIT Press.

Bannour, B. and Lapitre, A. (2020). Model checking of trickle-based IoT dissemination. In *2020 9th Mediterranean Conference on Embedded Computing (MECO)*, pages 1–6.

Bedford, A., Chong, S., Desharnais, J., Kozyri, E., and Tawbi, N. (2017). A progress-sensitive flow-sensitive inlined information-flow control monitor (ext. version). *Computers & Security*, 71:114–131.

Cardoza, C. (2016). Princeton tries to find out if your IoT devices are safe. Software Developpement Times (SDT). https://sdtimes.com/connected-devices/princeton-tries-to-find-out-are-your-iot-devices-safe/. Accessed: 2023-10-10.

Celik, Z. B., McDaniel, P., and Tan, G. (2018). Soteria: Automated IoT safety and security analysis. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 147–158.

Celik, Z. B., Tan, G., and McDaniel, P. D. (2019). IoTguard: Dynamic enforcement of security and safety policy in commodity IoT. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society.

Chen, G., Jiang, T., Wang, M., Tang, X., and Ji, W. (2020). Design and model checking of timed automata oriented architecture for internet of thing. *International Journal of Distributed Sensor Networks*, 16.

Chen, J., Diao, W., Zhao, Q., Zuo, C., Lin, Z., Wang, X., Lau, W., Sun, M., Yang, R., and Zhang, K. (2018). IoTfuzzer: Discovering memory corruptions in IoT through app-based fuzzing. In *Network and Distributed System Security Symposium*.

Chi, H., Zeng, Q., Du, X., and Yu, J. (2020). Cross-app interference threats in smart homes: Categorization, detection and handling. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 411–423.

Choe, Y. and Lee, M. (2018). Process model to predict nondeterministic behavior of IoT systems. In *PrOse@PoEM*.

Clements, A. A., Almakhdhub, N. S., Saab, K. S., Srivastava, P., Koo, J., Bagchi, S., and Payer, M. (2017). Protecting bare-metal embedded systems with privilege overlays. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 289–303.

Damodaran, A., Di Troia, F., Visaggio, C. A., Austin, T., and Stamp, M. (2017). A comparison of static, dynamic, and hybrid analysis for malware detection. *Journal of Computer Virology and Hacking Techniques*, 13.

Desharnais, J., Gupta, V., Jagadeesan, R., and Panangaden, P. (2002). Weak bisimulation is sound and complete for pctl*. In Brim, L., Jancar, P., Kretínský, M., and Kucera, A., editors, *CONCUR 2002 - Concurrency Theory, 13th International Conference*, volume 2421, pages 355–370. Springer.

Esquiagola, J., de Paula Costa, L. C., Calcina, P., Fedrecheski, G., and Zuffo, M. K. (2017). Performance test-

ing of an internet of things platform. In *International Conference on Internet of Things, Big Data and Security*.

Frappier, M., Fraikin, B., Chossart, R., Chane-Yack-Fa, R., and Ouenzar, M. (2010). Comparison of model checking tools for information systems. In Dong, J. S. and Zhu, H., editors, *Formal Methods and Software Engineering*, pages 581–596. Springer Berlin Heidelberg.

Gani, K., Bouet, M., Schneider, M., and Toumani, F. (2015). Using timed automata framework for modeling home care plans. In *2015 International Conference on Service Science (ICSS)*, pages 1–8.

Gerking, C., Schubert, D., and Bodden, E. (2018). Model checking the information flow security of real-time systems. In *Engineering Secure Software and Systems : 10th International Symposium, ESSoS 2018*, volume 10953, page 27–43.

Goguen, J. A. and Meseguer, J. (1982). Security policies and security models. In *1982 IEEE Symposium on Security and Privacy*, pages 11–11.

Klier, M., Moestue, L., Obermeier, A., and Widmann, T. (2024). Assessing completeness of IoT data: A novel probabilistic approach. *Business & Information Systems Engineering*, pages 1–18.

Kwiatkowska, M. and Parker, D. (2012). Advances in probabilistic model checking. In *Software Safety and Security*, pages 126–151. IOS Press.

Larsen, K. G., Pettersson, P., and Yi, W. (1995). Model-checking for real-time systems. In Reichel, H., editor, *Fundamentals of Computation Theory*, pages 62–88. Springer Berlin Heidelberg.

Lee, G. M. and Truong, N. B. (2016). A reputation and knowledge based trust service platform for trustworthy social internet of things. In *Innovations in Clouds, Internet and Networks (ICIN)*.

Li, S. and Xu, L. D. (2017). *Securing the Internet of Things*. Syngress Publishing, 1st edition.

Mafamane, R., Ouadou, M., Hassani, A., and Minaoui, K. (2021). Study of the heterogeneity problem in the internet of things and cloud computing integration. In *2020 10th International Symposium on Signal, Image, Video and Communications (ISIVC)*.

Mainuddin, M., Duan, Z., and Dong, Y. (2021). Network traffic characteristics of IoT devices in smart homes. In *2021 International Conference on Computer Communications and Networks (ICCCN)*, pages 1–11. IEEE.

McCall, M., Zeng, E., Shezan, F. H., Yang, M., Bauer, L., Bichhawat, A., Cobb, C., Jia, L., and Tian, Y. (2023). Towards usable security analysis tools for Trigger-Action programming. In *Nineteenth Symposium on Usable Privacy and Security (SOUPS 2023)*, pages 301–320, Anaheim, CA. USENIX Association.

Neagle, C. (2015). Smart refrigerator hack exposes gmail login credentials. https://www.networkworld.com/article/2976270/smart-refrigerator-hack-exposes-gmail-login-credentials.html. Accessed: 2023-10-10.

Ngo, T. and Nguyen, N. (2019). Secure information flow for IoT applications. *JOIV : International Journal on Informatics Visualization*, 3:192–197.

Nguyen, D. T., Song, C., Qian, Z., Krishnamurthy, S. V., Colbert, E. J. M., and Mcdaniel, P. (2018). IoTSan: fortifying the safety of IoT systems. *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies*.

Nunes, D., Zhang, P., and Sá Silva, J. (2015). A survey on human-in-the-loop applications towards an internet of all. *IEEE Communications Surveys & Tutorials*.

Park, M., Oh, H., and Lee, K. (2019). Security risk measurement for information leakage in iot-based smart homes from a situational awareness perspective. *Sensors*, 19(9).

Pradeep, P. and Kant, K. (2022). Conflict detection and resolution in IoT systems: A survey. *IoT*, 3(1):191–218.

Radovici, A., Culic, I., Rosner, D., and Oprea, F. (2020). A model for the remote deployment, update, and safe recovery for commercial sensor-based IoT systems. *Sensors*, 20(16).

Reddit (2018). Roomba sets off motion sensor at night? https://www.reddit.com/r/homeautomation/comments/92ahq3/roomba_sets_off_motion_sensor_at_night/. Accessed: 2023-10-10.

Sabelfeld, A. and Myers, A. C. (2003). Language-based information-flow security. *IEEE J. Sel. Areas Commun.*, 21:5–19.

Schneider, F. B. (2000). Enforceable security policies. *ACM Trans. Inf. Syst. Secur.*, 3(1):30–50.

Sha, L., Xiao, F., Chen, W., and Sun, J. (2018). Iiot-sidefender: Detecting and defense against the sensitive information leakage in industry iot. *World Wide Web*, 21:59–88.

Sobin, C. (2020). A survey on architecture, protocols and challenges in IoT. *Wireless Personal Communications*, 112(3):1383–1429.

Stevens, C., Alhanahnah, M., Yan, Q., and Bagheri, H. (2020). Comparing formal models of iot app coordination analysis. In *Proceedings of the 3rd ACM SIGSOFT International Workshop on Software Security from Design to Deployment*, page 3–10. Association for Computing Machinery.

Uppsala and Aalborg (2020). Uppaal. https://uppaal.org/. Accessed: 2023-10-18.

Wang, Q., Hassan, W. U., Bates, A., and Gunter, C. A. (2018). Fear and logging in the internet of things. In *Network and Distributed System Security Symposium*.

Yu, D., Li, P., Chen, Y., Ma, Y., and Chen, J. (2020). A time-efficient multi-protocol probe scheme for fine-grain IoT device identification. *Sensors*, 20.

Zetter, K. (2014). Hospital networks are leaking data, leaving critical devices vulnerable. https://www.wired.com/2014/06/hospital-networks-leaking-data/. Accessed: 2023-10-10.

Zhang, L. (2017). Fragmentation in IoT – one roadblock in IoT deployment. https://www.cleantech.com/fragmentation-in-iot-one-roadblock-in-iot-deployment/. Accessed: 2023-10-18.