Opacity in Internet of Things with Cloud Computing

Wen Zeng, Maciej Koutny, Paul Watson
School of Computing Science, Newcastle University
Newcastle upon Tyne NE1 7RU, U.K.
wen.zeng.wz@gmail.com, {maciej.koutny, paul.watson}@ncl.ac.uk

***Abstract*—Internet of Things (IoT) with Cloud Computing (CC) is a new paradigm incorporating a pervasive presence of a wide range of things/objects which can interact with each other and cooperate, creating new services and reaching common goals. This will lead to more intelligent smart environments in a wide range of applications. In this context, protecting the Internet of Things with Cloud Computing (IoTwCC) against interference, including service attacks and viruses, becomes paramount. In this paper, we introduce a transition system representation to capture the information flow in IoTwCCs, and then investigate the opacity of the information flow model. In addition, we introduce a threat model to describe the actions of the system, and propose entropy as a security metrics to quantify the amount of information related to a service that might be exposed to other users or adversaries. It turns out that the opacity of the system is affected by the availability of the services. As a result, the trade-off between opacity and service availability can be analyzed.**

Keywords*—*availability, entropy, information flow security, opacity, security metrics

I. Introduction

An increasing number of system developers use cloud technologies to provide IoT services. The key idea is that all edge devices and IoTs send information periodically to an application platform located in the cloud. The platform stores all the data and provides specialized interfaces that can be used by third parties to create their IoT applications. Moreover, there are platforms pursuing the idea of creating public and private clouds that can be deployed in a local environment. These platforms not only enable the existence of local intelligence but can also exchange information and services with external systems. Therefore, they can easily become instances of IoT.

Federated cloud systems (FCSs) increase the reliability and reduce the cost of computational support. However, the large number of services and data involved creates security risks due to the dynamic movement of the entities on the cloud system. As a result, the integration of IoT with cloud computing creates new security challenges that must be overcome in order to advance IoTwCC into the real world [1], [2].

A key role of information flow security is to ensure that information propagates throughout the execution environment without security violation; in particular, that no secure information is leaked to unauthorised subjects. It has been recognised that communication channels can create threats that can affect IoT entities. In IoTwCC, the information flow will follow a hierarchical pattern, as a central entity will receive information from every 'thing' [1]. Moreover, if an adversary targets an IoT enabled system in a distributed scenario, they might be able to retrieve processed information instead of raw data [1], which could help the attacker to access and control system services. Therefore, in this study, we will analyze the information flow security and opacity in IoTwCC.

There exists prior work addressing workflow security; for example, the flow-sensitive analysis of programs [3], [4], [5]. The paper [6] proposed to partition workflows over a set of available clouds in such a way that security requirements are met. Using Petri nets to model workflows, [7], [8], [9], [10] introduced flow sensitive security models to capture secure information flow in CC systems. However, the observation of the information flow, which leads to the notion of opacity, has not been analyzed there.

Opacity has been used to analyze the information-hiding properties of protocols and programs, and [11], [12], [13] proposed to use Petri nets to specify the opacity of information flow, and adapted opacity to labeled transition system. Moreover, [14], [15] studied the problem of information hiding in systems characterized by the presence of randomization and concurrency. Having said that, the existing studies have been strongly theoretical in the way they approached information hiding and opacity. In this paper, we make initial steps to bridge a gap between the theory of opacity and its practical application.

The paper is organized as follows. In Section II, a basic model for information flow in IoTwCC is presented. Then, the opacity of system is discussed in Section III. In Section IV, we look at the probabilistic behaviour of the system. Then in Section V, we propose entropy-based opacity metrics to evaluate the information flow security of IoTwCC. The trade-off between opacity and service availability is analyzed in Section VI. Section VII concludes the paper.

II. System Model

In this section, we describe a simple transition system model to capture the information flow in IoTwCC.

We assume that P and E are finite non-empty sets of respectively *platforms* (e.g., clouds, devices, computer, and 'things') and *entities* (or entity names). In what follows, an entity can have several different copies, and these copies may reside in different platforms. We further allow multiple copies of a single entity to be present in a single deployment platform. As a result, a *state* will be understood as any finite multiset s over the set $E \times P$. Thus, for example, if $s(a, p) = 4$ then we know that in the current state there are 4 copies of entity a residing in platform p. The elements of $E \times P$ will be referred to as *actual* entities. We will say that an actual entity (e, p) is *present* in a state s if $s(e, p) > 0$.

Definition 1 (ifm): An *information flow model* for IoTwCC is a pair

$$IFM = (\mathcal{A}, s_{init}) \ , \tag{1}$$

where $\mathcal{A}$ is a finite set of actions and s_{init} is an initial state. It is assumed that each action is a pair

$$\phi = (in, out) \tag{2}$$

such that its components

$$\begin{aligned} in &= (e_1@p_1, \ldots, e_k@p_k) \ \text{ and} \\ out &= (e_{k+1}@p_{k+1}, \ldots, e_{k+m}@p_{k+m}) \end{aligned}$$

specify finite tuples of entity-platform pairs. □

One might easily extend the class of allowed action types to include, e.g., checking of the presence of certain entities.

Definition 2 (single action execution): An action ϕ as in (2) is *enabled* at state s if

$$\phi^{in} = \{(e_1, p_1), \ldots, (e_k, p_k)\} \leq s \ .$$

Such an action can then be *executed* leading to a new state

$$s' = s - \phi^{in} + \phi^{out} \ ,$$

where $\phi^{out} = \{(e_{k+1}, p_{k+1}), \ldots, (e_{k+m}, p_{k+m})\}$ while '$-$' and '$+$' denote multiset subtraction and addition, respectively.

We denote this by $s \xrightarrow{\phi}_{\diamond} s'$. □

The above definition captures the enabling and execution of a single action. The next definition lifts this to any group of actions executed simultaneously.

Definition 3 (multiset action execution): Consider a multiset $K = \{\phi_1 \ldots, \phi_n\}$ such that each ϕ_i is an action enabled at state s and, moreover,

$$K^{in} = (\phi_1)^{in} + \ldots + (\phi_n)^{in} \leq s \ .$$

Then K can then be *executed* leading to a new state

$$s' = s - K^{in} + K^{out} \ ,$$

where $K^{out} = (\phi_1)^{out} + \ldots + (\phi_n)^{out}$.

We denote this by $s \xrightarrow{K}_{\diamond} s'$. □

Definition 4 (reachable states): The set of reachable states of the information flow model as in (1) is the minimal set of states RS containing s_{init} and such that if $s \in RS$ and $s \xrightarrow{K}_{\diamond} s'$, for some K, then $s' \in RS$. □

We have provided basic notions related to the syntax and operational semantics of an information flow model. It allows straightforward capture of various notion of information flow for IoTwCC.

The inherent complexity of IoT is due to the fact that multiple heterogeneous entities located in different contexts can exchange information with each other [1]. In such service-based computing system, information sharing means that the behaviour of one user may appear visible to other users or adversaries, and observations of such behaviours can potentially help adversaries to build covert channels. Therefore, it is necessary to analyze the information leakage. Here we consider using opacity [11], [12], [13] as a promising technique for analyzing information flow security in IoTwCC.

III. Opacity in IoTwCC

In this section, we will show how opacity can be used to analyze the security property of IoTwCC systems.

Opacity is a uniform approach for describing security properties expressed as predicates [13]. A predicate is opaque if an observer of the system is unable to determine the truth of the predicate in a given run of the system. In this section, we will discuss one of the versions of opacity in the context of workflows executed on IoTwCC systems.

Let $IFM = (\mathcal{A}, s_{init})$ be a information flow model as in (1). A run of IFM is a finite sequence

$$\xi = \kappa_1 \ldots \kappa_n \quad (n \geq 0) \tag{3}$$

such that there are states $s_{init} = s_1, \ldots, s_{n+1}$ satiafying $s_i \xrightarrow{\kappa_i}_{\diamond} s_{i+1}$, for $i = 1, \ldots, n$. The set of all runs of IFM will be denoted by $RUN(IFM)$.

To model the different capabilities for observing the system modeled by IFM one can use *observation functions*:

$$obs : RUN(IFM) \to Obs^*$$

where Obs be a set of *observables*. In what follows, we consider a *state observation function* obs for which there is a map obs' associating $obs'(K) \in Obs \cup \{\epsilon\}$ with every K as in Definition 3, in such a way that

$$obs(\xi) = obs'(\kappa_1) \ldots obs'(\kappa_n)$$

for every run $\xi = \kappa_1 \ldots \kappa_n$ in $RUN(IFM)$.

Given the observation function obs, we are now interested in whether an observer can establish a property γ (a predicate over system runs) for a run of IFM having only access to the result of the observation function. As one can identify γ with its characteristic set, i.e., the set of all those runs for which it holds, we want to find out whether the fact that the underlying run belongs to $\gamma \subseteq RUN(IFM)$ can be deduced by the observer on the basis of an observed execution of the system. Moreover, we are interested in the *final* opacity predicate γ_Z, defined as the set of all the runs ξ as in (3) satisfying $s_{n+1} \in Z$, for some set of states Z [13]. Intuitively, this means that we are interested in finding out whether an observed run of the system represented by IFM ended in one of secret (or sensitive) states belonging to Z. Note that we are not interested in establishing whether the underlying run does not belong to γ; to do this, we would consider the property $\bar{\gamma} = RUN(IFM) \setminus \gamma$.

We then say that γ is *opaque* w.r.t. obs if, for every run $\xi \in \gamma$, there is another run $\xi \in \bar{\gamma}$ such that $obs(\xi) = obs(\xi')$. In other words, if all runs in γ are covered by the runs in $\bar{\gamma}$:

$$obs(\gamma) \subseteq obs(\bar{\gamma}) \ .$$

A. Case study

As a case study adapted to our purposes, we will use a medical research application [16] in which data from a set of patients' heart rate monitors is analyzed, and then the results are sent to the things, as illustrated in Figure 1. Informally, the process can be described as follows:

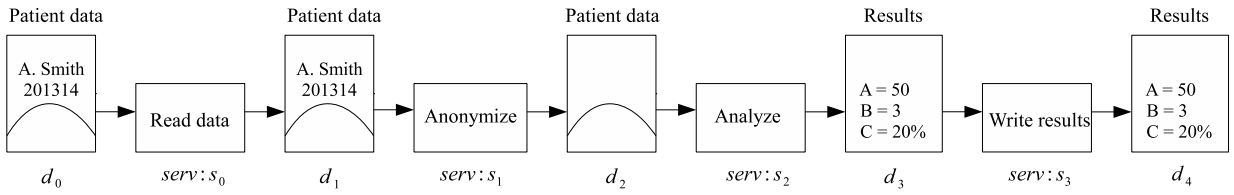

Fig. 1. The medical research application example, which includes four services and five finds of data.

- A patients' data is sent through the local network by $user$. The data (d_0) is a file with a header identifying the patient (name and patient number), followed by a set of heart rate data recoded over a period of time.
- A service ($serv{:}s_0$) reads the data, and changes the name of the data into (d_1), then sends the data to service ($serv{:}s_1$).
- A service ($serv{:}s_1$) strips off the header, leaving only the heart rate data (d_2).
- A third service ($serv{:}s_2$) analyzes the heart rate data, and produces results (d_3).
- Finally, service ($serv{:}s_3$) sends the data (d_4) to $user/medical\ device$, and then the 'medical device' can provide appropriate service for the patient based on the received data.

Analyzing the heart rate data ($serv{:}s_3$) is costly and would benefit from a cheap, scalable resources that are available on public clouds. However, considering that storing medical records on a public cloud can breach confidentiality, some organizations prefer to deploy the whole workflow on a secure private cloud. Such a policy may overstretch the limited resources available on the private cloud, resulting in degraded performance and negative impact on other applications. To address this problem, the partitioning of the application between a private cloud and a public cloud could provide a better solution.

In our case study, we consider an integration of IoT with CC. We use two clouds (X, and Y), one local network LN, and a number of processes, which together form a *medical research* application. The proposed workflow operates on sensitive medical data processed on the private cloud, and anonymised data that can be deployed on the public cloud. Cloud *X* hosts services $serv{:}s_0$ and $serv{:}s_1$. Cloud *Y* hosts two service providers: *service provider 1* includes $serv{:}s_2^1$ and $serv{:}s_3^1$; and *service provider 2* includes $serv{:}s_2^2$ and $serv{:}s_3^2$.

Figure 2 shows the basic structure of the execution scenario for the medical research application. The generic behaviour of medical research application is shown in Table I. It starts with a data sent from $user$ to $serv{:}s_0$, through a local network. The data is forwarded to service $serv{:}s_1$. Service $serv{:}s_1$ then selects one of the two providers, *service provider 1* or *service provider 2*, and then sends d_2 to the selected service providers. After receiving the data, $serv{:}s_2^1$ or $serv{:}s_2^2$ produces d_3 and sends it to $serv{:}s_3^1$ or $serv{:}s_3^2$, respectively. Finally, $serv{:}s_3^1$ and $serv{:}s_3^2$ sends d_4 to $user$ and $medical\ device$. Crucially, an observer of the system is not allowed to discover the identity of the selected provider.

TABLE I. THE SEQUENCE OF INTERACTIONS BETWEEN THE COMPONENTS OF THE MEDICAL RESEARCH APPLICATION SYSTEM.

Actions	Entities	Sender	Receiver
κ_1	d_0	$user$	$serv{:}s_0$
κ_2	d_1	$serv{:}s_0$	$serv{:}s_1$
κ_3^i	d_2	$serv{:}s_1$	$serv{:}s_2^i$ (i=1,2)
κ_4^i	d_3	$serv{:}s_2^i$ (i=1,2)	$serv{:}s_3^i$ (i=1,2)
κ_5^i	d_4	$serv{:}s_3^i$ (i=1,2)	$user$
κ_6^i	d_4	$serv{:}s_3^i$ (i=1,2)	$medical\ device$

1) Case one: We assume that no provider is discriminated against. Moreover, messages communicated between the clouds *X*, *Y* and local network are visible, and messages inside the clouds are invisible. However, the observer has no means of detecting their content (but can observe the specific cloud from which a message originated or was sent to). This can be captured by the following observation function:

$$\begin{array}{lcllcl} obs'(\kappa_1) & = & a & obs'(\kappa_2) & = & \epsilon \\ obs'(\kappa_3^1) & = & b & obs'(\kappa_3^2) & = & b \\ obs'(\kappa_4^1) & = & \epsilon & obs'(\kappa_4^2) & = & \epsilon \\ obs'(\kappa_5^1) & = & d & obs'(\kappa_5^2) & = & d \\ obs'(\kappa_6^1) & = & e & obs'(\kappa_6^2) & = & e \end{array}$$

Using opacity, we may show that visible interactions do not reveal the identity of the provider supplying the service. To see this, we consider a property γ consisting of all execution scenarios where the first provider supplied the services, i.e., executions of the following form:

$$\begin{array}{lcl} \xi_1 & = & \kappa_1\kappa_2\kappa_3^1\kappa_4^1\kappa_5^1\kappa_6^1 \\ \xi_2 & = & \kappa_1\kappa_2\kappa_3^1\kappa_4^1\kappa_6^1\kappa_5^1 \end{array}$$

The set of observations generated is therefore given by $obs(\gamma) = \{obs(\xi_i) : i = 1, 2\}$, where

$$\begin{array}{lcl} obs(\xi_1) & = & \{abde\} \\ obs(\xi_2) & = & \{abed\} \end{array}$$

We then note that $\bar{\gamma}$ comprises, among others, executions of the following kind:

$$\begin{array}{lcl} \xi_1 & = & \kappa_1\kappa_2\kappa_3^2\kappa_4^2\kappa_5^2\kappa_6^2 \\ \xi_2 & = & \kappa_1\kappa_2\kappa_3^2\kappa_4^2\kappa_6^2\kappa_5^2 \end{array}$$

The set of observations generated therefore satisfies $obs(\bar{\gamma}) \supseteq \{obs(\xi_i) : i = 1, 2\}$, where

$$\begin{array}{lcl} obs(\xi_1) & = & \{abde\} \\ obs(\xi_2) & = & \{abed\} \end{array}$$

Hence $obs(\gamma) \subseteq obs(\bar{\gamma})$, and so γ is an opaque property. As a result, it is never possible to say for sure that it was the

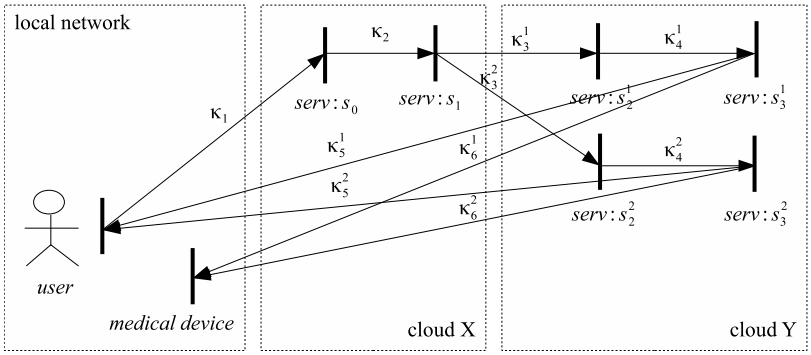

Fig. 2. Information flow in a cloud based *medical research* application.

first provider who supplied the service. Since the argument is completely symmetric, we can conclude that the identity of providers is kept secret.

2) Case two: We assume that cloud *Y* is public and cloud *X* is private. Messages communicated between the clouds *X*, *Y* and the local network are visible, messages inside of cloud *Y* are visible, messages inside of cloud *X* are invisible. This can be captured by the following (static) observation function:

$$\begin{array}{lcllcl} obs'(\kappa_1) & = & a & obs'(\kappa_2) & = & \epsilon \\ obs'(\kappa_3^1) & = & b & obs'(\kappa_3^2) & = & c \\ obs'(\kappa_4^1) & = & d & obs'(\kappa_4^2) & = & e \\ obs'(\kappa_5^1) & = & f & obs'(\kappa_5^2) & = & g \\ obs'(\kappa_6^1) & = & h & obs'(\kappa_6^2) & = & k \end{array}$$

We consider the property γ consisting of all execution scenarios where the first provider supplied the services (the executions are the same as the first case). The set of observations they generate is given by $obs(\gamma) = \{obs(\xi_i) : i = 1, 2\}$, where

$$\begin{array}{lcl} obs(\xi_1) & = & \{abdfh\} \\ obs(\xi_2) & = & \{abdhf\} \end{array}$$

When the second provider supplied the services, the set of observations generated is $obs(\bar{\gamma}) = \{obs(\xi_i) : i = 1, 2\}$, where

$$\begin{array}{lcl} obs(\xi_1) & = & \{acegk\} \\ obs(\xi_2) & = & \{acekg\} \end{array}$$

Hence $obs(\gamma) \backslash obs(\bar{\gamma}) \neq \emptyset$, and so γ is not an opaque property. Thus, it is now possible to say that it was the first provider who supplied the services.

From these two case studies we can see that an observation cannot establish a predicate if for any run of the system in which the predicate is true, there is a run for which the predicate is false, and the two runs are equivalent under the defined observation function.

In this section, we only considered how security polices can affect the information flow security. We have not yet taken into account the likelihood of violating the opacity requirement; instead, we simply reported whether a given system is opaque or not. This yes/no outcome might be not satisfactory in practice, especially when system behaviours have unequal likelihood of occurring. In the case where the probability of one run is significantly higher than the probability of another one, the observer may have good reasons to believe that the predicate is none the less true. Therefore, in the next section, we will analyze the probabilistic behaviour of the system and measure the opacity of security properties.

IV. Probabilistic Behaviours

We now consider the probabilistic opacity in cloud systems which will allow us to reason about quantitative characteristics of security properties.

Firstly, we consider the probability distributions on random variables of execution traces. Let D be a finite set with a discrete probability distribution. A discrete random variable X is a surjective function which maps the elements of D (events) to values of a countable set (e.g., integers), with each value in the range having probability greater than zero, and $\mathcal{R}$ is the finite range of X, i.e., $X : D \to \mathcal{R}(D)$ [17]. For each $d \in D$, we write $p(d)$ for its probability, i.e., $0 < p(d) \leq 1$ and $\sum_{d \in D} p(d) = 1$.

The system can be considered as a communication channel, and a discrete random variable X constructed to model a finite set of possible traces performed by the end users/entities during their interactions with the system. Each trace is a finite sequence of actions performed by the end user/entity, i.e., $\xi = \kappa_1 \ldots \kappa_k$, which captures the way in which an end user reaches the finial action κ_n from the initial action κ_1 when requesting a specific service. In addition, each trace is associated with a positive probability, and the sum of the probabilities of all possible traces is 1.

In this study, we require that the number of traces of the system is finite, and $\xi, \ldots, \xi_n$ are all the finite traces generated by the system. In order to consider the probabilistic behaviour and opacity, we assume that for the traces in the system we have $\sum_{i=1}^{n} p(\xi_i) = 1$.

Following the definition of *observation function* in Section III, we then have a number of distinct observed traces, $\psi_1 \ldots, \psi_m$, such that $\{\psi_1 \ldots, \psi_m\} = obs(\{\xi, \ldots, \xi_n\})$. The probability of each observed trace ψ_i is $p(\psi_i) = \sum_{\xi_j \in obs^{-1}(\psi_i)} p(\xi_j)$. Clearly, the observed traces form a distribution as $\sum_{i=1}^{m} p(\psi_i) = 1$.

We now consider that an IoTwCC system includes:

- a finite set of *actions*, $\mathcal{A}$, modelling the interactions between the server and the end users/entities;
- a finite set of *observables*, Obs, during the interactions between the end users/entities and the system;

- a set of *actual traces*, $\mathcal{T}$, modelling the sequences of actions performed by the end users/entities during interacting with the system;
- a set of *observed traces*, $\mathcal{O}$, which can be obtained from the actual traces;
- a family of *probability distributions*, $\mu_{\mathcal{T}}$, each of which is built on all actual traces of an entity interacting with the server; and
- a family of *probability distributions*, $\mu_{\mathcal{O}}$, each of which is built on all observed traces obtained from the actual traces.

It is assumed that the random variables on the traces are obtained by *repeated experiments*. For any specific end user/entity, let all possible traces of a user/entity requesting a service located in the system be denoted by

$$\mathcal{T} = \{\xi_i \mid 1 \leq i \leq n\} \textit{ and } \sum_{i=1}^{n} p(\xi_i) = 1 \ .$$

The observed traces can be viewed as projections of the actual traces, thus, we have

$$\mathcal{O} = \{\psi_i \mid 1 \leq i \leq m\} \textit{ and } \sum_{i=1}^{m} p(\psi_i) = 1 \ .$$

Therefore, information about the actual users' behaviour can be partially deduced by observing system traces. Under repeated observation of actual traces, one can derive the probability distribution for the projections of the possible traces for end users/entities.

In our threat model, we assume that there are two types of actions, *hidden* and *observable* actions, which are related to *high* security level and *low* security level, respectively. Actions labeled *hidden* are confidential and invisible to the adversaries, and actions labeled *observable* are public and observable to the adversaries. The classification can be based on the security preserving mechanisms or policies applied in the computing system. Therefore, the actions can be described as the union of two disjoint sets:

$$\mathcal{A} = high \uplus low$$

For each trace, some part of it may be hidden and some part observable, which makes some traces equivalent to the others when only considering observable actions. Therefore, adversaries can derive some confidential information by building sets of equivalence classes from the observations.

V. Entropy-based Notions of Opacity

In this section, entropy is used to measure the opacity of the system.

A. Shannon's Measure of Entropy

Shannon's information theory [18] can be used to quantify the amount of information a system may leak and the way in which this depends on the distribution of inputs.

Definition 5 (Shannon entropy): Let X be a random variable, x range over the set of values which X may take, and $p(x)$ denotes the probability that X takes the value x. The entropy of a discrete random variable X is denoted by $\mathcal{H}(X)$ and defined by $\mathcal{H}(X) = \sum_x p(x) \log_2 \frac{1}{p(x)}$. □

The entropy measures the average information content of a set of events. In the extreme case of an event with probability 1, the entropy is 0. On the other hand, if the probability distribution is uniform, the entropy is maximal.

Definition 6 (conditional entropy): The conditional entropy $\mathcal{H}(X|Y)$ measures the uncertainty about X, given the knowledge of $Y = y$. It is defined as: $\mathcal{H}(X|Y) = \mathcal{H}(X,Y) - \mathcal{H}(Y)$ where the joint entropy $\mathcal{H}(X,Y)$ of a pair of discrete random variables (X,Y) with a joint distribution $p(x,y)$ is defied as $\mathcal{H}(X,Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y) \frac{1}{\log_2 p(x,y)}$. □

Given two random variables, X and Y, the conditional entropy captures dependencies between random variables, when the knowledge of one may change the information about the other. If $\mathcal{H}(X|Y) = 0$, there is no uncertainty about X knowing Y; and if X and Y are independent (i.e., $p(x,y) = p(x)p(y)$), then $\mathcal{H}(X|Y) = \mathcal{H}(X)$, meaning that the knowledge of Y does not change the uncertainty on X.

The concept of mutual information is a measure of the amount of information that one random variable contains about another random variable.

Definition 7 (mutual information): Let $p(x,y)$ be the joint distribution of two random variables, X and Y. The mutual information $\mathcal{I}(X;Y)$ between X and Y is given by: $\mathcal{I}(X;Y) = \mathcal{H}(X) - \mathcal{H}(X|Y) = \mathcal{H}(Y) - \mathcal{H}(Y|X)$. □

If X and Y are independent, then knowing X does not give any information about Y and vice versa, so their mutual information is zero. At the other extreme, if X and Y are identical then all information conveyed by X is shared with Y. As a result, in the case of identity, the mutual information is the same as the uncertainty contained in X (or Y) alone, namely the entropy of X (or Y as identical X and Y have equal entropy).

Channel capacity is defined as the maximum mutual information, $C = max_{p(x)}\mathcal{I}(X;Y)$, where the maximum is taken over all possible input distributions $p(x)$. The channel is said to be *memoryless* if the probability distribution of the output depends only on the input at that time and is conditionally independent of previous channel inputs or outputs.

B. Entropy-based Opacity

We consider possible traces of an end user interacting with a service provider in the system as random variables, and define the quantity of observations of the computing system by using the concept of mutual information between the actual traces and observations.

Definition 8: Consider any specific user u requesting service s. The quantity of entropy-based information (uncertainty) loss to user u in the computing system is defined as:

$$\mathcal{I} = \mathcal{I}(T_{u,s}; O_{u,s}) = \mathcal{H}(T_{u,s}) - \mathcal{H}(T_{u,s}|O_{u,s})$$

where $T_{u,s}$ and $O_{u,s}$ denote the random variables for the user's actual traces $\mathcal{T}_{u,s}$ and observation traces $\mathcal{O}_{u,s}$, respectively; $\mathcal{H}(T_{u,s})$ is the entropy (uncertainty measurement) of traces

in $\mathcal{T}_{u,s}$, and $\mathcal{H}(T_{u,s}|O_{u,s})$ is the conditional entropy of $T_{u,s}$, given the observation $O_{u,s}$ (the remaining uncertainty $\mathcal{T}_{u,s}$ after observing $\mathcal{O}_{u,s}$). This yields entropy-based $\overline{\pi}_{\mathcal{I}}$-opacity. □

1) Case Study: Consider again the case of Section III-A. There are four different actual traces that can be generated by the system and we assume $p(\xi_i) = \frac{1}{4}$, for all i.

In the first case, there are two different observed traces, $obs = \{\delta_1, \delta_2\}$, and we have: $p(\delta_1) = p(\delta_2) = \frac{1}{2}$ For $\mathcal{T}_{u,s} = \{\xi_1\}$ and $\mathcal{O}_{u,s} = \{\delta_1\}$, we then obtain

$$\begin{aligned}\mathcal{I}(\xi_1;\delta_1) &= \mathcal{H}(\xi_1) - \mathcal{H}(\xi_1|\delta_1)\\ &= 4 \cdot \tfrac{1}{4}\log_2 4 - (2 \cdot \tfrac{1}{2}\log_2 2) = 1\end{aligned}$$

The mutual information between ξ_1 and δ_1 is therefore 1, yielding $\overline{\pi}_1$-opacity.

In the second case, there are four different observed traces, $obs = \{\delta_1, \delta_2, \delta_3, \delta_4\}$, and we have: $p(\delta_1) = p(\delta_2) = p(\delta_3) = p(\delta_4) = \frac{1}{4}$. We then obtain:

$$\begin{aligned}\mathcal{I}(\xi_1;\delta_1) &= \mathcal{H}(\xi_1) - \mathcal{H}(\xi_1|\delta_1)\\ &= 4 \cdot \tfrac{1}{4}\log_2 4 - (4 \cdot \tfrac{1}{4}\log_2 4) = 0\end{aligned}$$

The mutual information between ξ_1 and δ_1 is 0, yielding $\overline{\pi}_0$-opacity.

IoT is a hugely demanding environment due to the potentially unbounded number of things (resources and subjects). As the above discussion focused on the case of one end user interacting with the system, it is still necessary to define the security measurement for all end users in the system.

C. Channel Capacity

We now define opacity measurement based on the concept of channel capacity. Let us consider the security preserving mechanisms providing secure communication channels for clients and services in the information theoretical sense. The channel's capacity is the maximum mutual information between T and O over all possible end users/entities w.r.t. to requesting a service s, where T and O respectively denote the random variables of $\mathcal{T}$ (the set of actual traces) and $\mathcal{O}$ (the set of observed traces).

Definition 9: Channel capacity is defined as

$$C = max_{u \in U}\ \mathcal{I}(T_u; O_u)\ ,$$

yielding $\widehat{\pi}_C$-opacity. □

1) Case Study: Consider the second case of Section III-A again. We assume there are two end users u_1 and u_2 sending (κ_1) data d_0 to $serv{:}s_0$. The sum of the probabilities of all traces performed by all the users is 1. The probability of all possible traces of u_1 is $\frac{2}{3}$, and that of u_2 is $\frac{1}{3}$. The possible actual traces of each user and conditional probabilities of each user's traces are as follows:

End Users	Actual Traces	Probability
$\frac{2}{3}.u_1$	$\xi_1 = \kappa_1\kappa_2\kappa_3^1\kappa_4^1\kappa_5^1\kappa_6^1$	$\frac{1}{6}$
	$\xi_2 = \kappa_1\kappa_2\kappa_3^1\kappa_4^1\kappa_6^1\kappa_5^1$	$\frac{1}{6}$
	$\xi_3 = \kappa_1\kappa_2\kappa_3^2\kappa_4^2\kappa_5^2\kappa_6^2$	$\frac{1}{3}$
	$\xi_4 = \kappa_1\kappa_2\kappa_3^2\kappa_4^2\kappa_6^2\kappa_5^2$	$\frac{1}{3}$
$\frac{1}{3}.u_2$	$\xi_1 = \kappa_1\kappa_2\kappa_3^1\kappa_4^1\kappa_5^1\kappa_6^1$	$\frac{1}{8}$
	$\xi_2 = \kappa_1\kappa_2\kappa_3^1\kappa_4^1\kappa_6^1\kappa_5^1$	$\frac{1}{8}$
	$\xi_3 = \kappa_1\kappa_2\kappa_3^2\kappa_4^2\kappa_5^2\kappa_6^2$	$\frac{3}{8}$
	$\xi_4 = \kappa_1\kappa_2\kappa_3^2\kappa_4^2\kappa_6^2\kappa_5^2$	$\frac{3}{8}$

The observations are therefore as follows:

End Users	Observed Traces	Probability
$\frac{2}{3}.u_1$	$obs(\xi_1) = abdfh$	$\frac{1}{6}$
	$obs(\xi_2) = abdhf$	$\frac{1}{6}$
	$obs(\xi_3) = acegk$	$\frac{1}{3}$
	$obs(\xi_4) = acekg$	$\frac{1}{3}$
$\frac{1}{3}.u_2$	$obs(\xi_1) = abdfh$	$\frac{1}{8}$
	$obs(\xi_2) = abdhf$	$\frac{1}{8}$
	$obs(\xi_3) = acegk$	$\frac{3}{8}$
	$obs(\xi_4) = acekg$	$\frac{3}{8}$

The entropy-based information loss due to the behaviour of u_1 is $\mathcal{I}_{u_1}(T_{u_1}; O_{u_1}) = \mathcal{H}(\frac{1}{6}, \frac{1}{6}, \frac{1}{3}, \frac{1}{3}) = 1.9183$.

The entropy-based information loss due to the behaviour of u_2 is $\mathcal{I}_{u_2}(T_{u_2}; O_{u_2}) = \mathcal{H}(\frac{1}{8}, \frac{1}{8}, \frac{3}{8}, \frac{3}{8}) = 1.8113$.

The channel capacity of the system is therefore given by $C = max_{u \in U}\ \mathcal{I}(T_u; O_u) = 1.9183$, yielding $\widehat{\pi}_{1.9183}$-opacity.

Note that different security preserving mechanisms or policies might produce different observations. Given a computing system, it is necessary to study the relations among the security policies applied by the system from the point view of the resulting degree of security.

VI. Trade-off between opacity and service availability

We continue the case study of Section V-C, assuming that if *service provider 1* breaks down then *service provider 2* is used as a replacement. If the availability of *service provider 1* is x, we obtain the following:

End Users	Observed Traces	Probability
$\frac{2}{3}.u_1$	$obs(\xi_1) = abdfh$	$\frac{1}{6} \cdot x$
	$obs(\xi_2) = abdhf$	$\frac{1}{6} \cdot x$
	$obs(\xi_3) = acegk$	$\frac{1}{3} + \frac{1}{6} \cdot (1 - x)$
	$obs(\xi_4) = acekg$	$\frac{1}{3} + \frac{1}{6} \cdot (1 - x)$
$\frac{1}{3}.u_2$	$obs(\xi_1) = abdfh$	$\frac{1}{8} \cdot x$
	$obs(\xi_2) = abdhf$	$\frac{1}{8} \cdot x$
	$obs(\xi_3) = acegk$	$\frac{3}{8} + \frac{1}{8} \cdot (1 - x)$
	$obs(\xi_4) = acekg$	$\frac{3}{8} + \frac{1}{8} \cdot (1 - x)$

We therefore have:

$$\mathcal{I}_{u_1}(T_{u_1};O_{u_1})=\mathcal{H}(\frac{1}{6}\cdot x,\frac{1}{6}\cdot x,\frac{1}{3}+\frac{1}{6}\cdot(1-x),\frac{1}{3}+\frac{1}{6}\cdot(1-x))$$

$$\mathcal{I}_{u_2}(T_{u_2};O_{u_2})=\mathcal{H}(\frac{1}{8}\cdot x,\frac{1}{8}\cdot x,\frac{3}{8}+\frac{1}{8}\cdot(1-x),\frac{3}{8}+\frac{1}{8}\cdot(1-x))$$

Figure 3 shows the channel capacity based opacity of the system against the availability of *service provider 1*. Since in real life distributed systems the number of the parameters is quite high, in our study the following ones were kept fixed: the probability of all possible traces of u_1 was set to $\frac{2}{3}$, and that of u_2 to $\frac{1}{3}$.

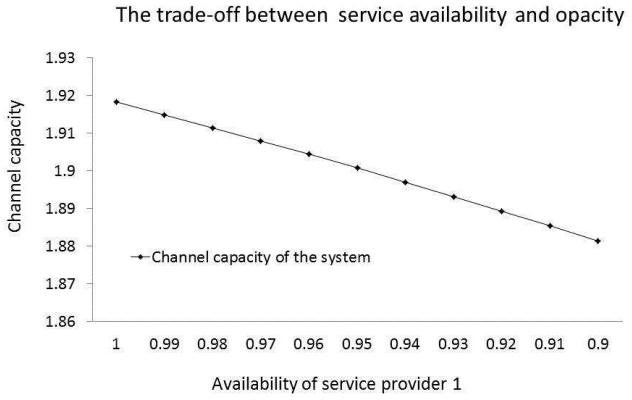

Fig. 3. The trade-off between service availability and opacity.

The graph of Figure 3 clearly indicates that a system with high level of availability of *service provider 1* would suffer from low degree of opacity or, in other words, high information leakage.

VII. Conclusions

In this paper, we proposed a simple model for tracking the information flow in IoTwCC systems. Opacity, as a promising technique for unifying a range of security properties, has been used to analyze the behaviour of a system. A medical research application example was given to discuss how observing a system's behaviour can help attacker to infer information about the system. Entropy was introduced to quantify the amount of information related to a service that might be exposed to the attackers. Moreover, channel capacity was used to measure the maximum of information loss among all end users. Finally, we investigated how the opacity of a system is affected by the availability of services.

The study presented in this paper can help to track and control the secure information flow in complex computing systems, and analyze the impact of different resource allocation strategies.

VIII. Acknowledgements

We would like to thank the referees for their comments and useful suggestions. This research was supported by the 973 Program Grant 2010CB328102, NSFC Grant 61133001, and EPSRC UNCOVER project.

References

[1] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," *Computer Networks*, vol. 57, no. 10, pp. 2266 – 2279, 2013, towards a Science of Cyber Security Security and Identity Architecture for the Future Internet.

[2] D. Miorandi, S. Sicari, F. D. Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497 – 1516, 2012.

[3] G. Smith, "A new type system for secure information flow," in *In CSFW14*. IEEE Computer Society Press, 2001, pp. 115–125.

[4] S. Hunt and D. Sands, "On flow-sensitive security types," in *Conference Record of the 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ser. POPL '06. New York, NY, USA: ACM, 2006, pp. 79–90.

[5] A. Russo, A. Sabelfeld, and A. Chudnov, "Tracking information flow in dynamic tree structures," in *Proceedings of the 14th European Conference on Research in Computer Security*, ser. ESORICS'09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 86–103.

[6] P. Watson, "A multi-level security model for partitioning workflows over federated clouds," *Journal of Cloud Computing*, vol. 1, no. 1, pp. 1–15, 2012.

[7] K. Knorr, "Dynamic access control through petri net workflows," in *Proceedings of the 16th Annual Computer Security Applications Conference*, ser. ACSAC '00, 2000, pp. 159–167.

[8] ——, "Multilevel security and information flow in petri net workflows," in *9th International Conference on Telecommunication Systems - Modeling and Analysis, Special Session on Security Aspects of Telecommunication Systems*, 2001.

[9] W. Zeng, C. Mu, M. Koutny, and P. Watson, "A flow sensitive security model for cloud computing systems," *CoRR*, vol. abs/1404.7760, 2014.

[10] W. Zeng, M. Koutny, and P. Watson, "Verifying secure information flow in federated clouds," in *IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom)*, 2014.

[11] J. Bryans, M. Koutny, and P. Y. A. Ryan, "Modelling opacity using petri nets," *Electron. Notes Theor. Comput. Sci.*, vol. 121, pp. 101–115, February 2005.

[12] J. Bryans, M. Koutny, L. Mazar, and P. Ryan, "Opacity generalised to transition systems," in *Formal Aspects in Security and Trust*, ser. Lecture Notes in Computer Science, 2006, vol. 3866, pp. 81–95.

[13] J. Bryans, M. Koutny, L. Mazaré, and P. Y. A. Ryan, "Opacity generalised to transition systems," *Int. J. Inf. Sec.*, vol. 7, no. 6, pp. 421–435, 2008.

[14] M. E. Andrés, C. Palamidessi, P. Van Rossum, and G. Smith, "Computing the leakage of information-hiding systems," in *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2010, pp. 373–389.

[15] M. E. Andrés, C. Palamidessi, P. van Rossum, and A. Sokolova, "Information hiding in probabilistic concurrent systems." in *QEST'10*, 2010, pp. 17–26.

[16] P. Watson and M. Little, "Multi-level security for deploying distributed applications on clouds, devices and things," in *IEEE 6th International Conference on Cloud Computing Technology and Science, CloudCom 2014, Singapore, December 15-18, 2014*, 2014, pp. 380–385.

[17] A. Renyi, *Probability Theory*. North-Holland Publishing Company, 1970.

[18] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.