Security Analysis of IoT systems using Attack Trees

Delphine Beaulaton[1], Najah Ben Said[3], Ioana Cristescu[2], and Salah Sadou[1]

[1] Univ. South Brittany, Irisa - France
[2] INRIA - France
[3] Thales SIX-GTS - Palaiseau

Abstract. Attack trees are graphical representations of the different scenarios that can lead to a security failure. In combination with model checking, attack trees are useful to quantitatively analyse the security of a system. Such analysis can help in the design phase of a system to decide how and where to modify the system in order to meet some security specifications.
In this paper we propose a security-based framework for modeling IoT systems where attack trees are defined alongside the model. A malicious entity uses the attack tree to exploit the vulnerabilities of the system. Successful attacks can be *rare events* in the system's execution, in which case they are hard to detect with usual model checking techniques. Hence, we use *importance splitting* as a statistical model checking technique for rare events. This technique requires a decomposition of an attack into sub parts, similarly to an attack tree. We argue that therefore, importance splitting is well suited, and benefits, from our modeling framework. We implemented our approach in a tool-set and verified its effectiveness by running a set of experiments over a real-word example.

1 Introduction

The Internet of Things (IoT) is a rapidly emerging paradigm that provides a practical and easier way for users to manage and control a large variety of objects interacting over the Internet. However, IoT systems involve heterogeneous devices that are connected to a shared network and that carry critical tasks, and hence, are targets for malicious users. Vulnerabilities are discovered in opportunistic manner since security has mostly an ad-hoc treatment. Therefore, developing a systematic mechanism that considers security aspects at an early stage of system design helps detecting and preventing attacks.

Formal security analysis usually target systems with well-defined properties and specific implementation. For IoT systems, we have to find an appropriate abstraction level that is applicable to different implementations. Security issues can occur at different levels in a system, for example in the computation nodes, in the communication protocols or at the storage level. *Attack trees* [12, 15] are intuitive and practical formal methods to identify and analyze attacks. As their name suggests, attacks are modeled as trees, where the leaves represent elementary steps needed for the attack, and the root represents a *successful attack*. The internal nodes are of two types, indicating whether all the sub-goals (an AND node) or one of the sub-goals (an OR node) must be achieved in order to accomplish the main goal. Moreover, attack trees can express security issues of different nature in an uniform way. Hence, combining both formal analysis and attack trees helps to track and monitor the entire system in order to detect security breaches.

In this paper we present a framework to formally model IoT systems and analyse them using attack trees. In the formal modeling language we introduce, IoT systems are represented as a set of entities that communicate with each other if some verification on their identity holds. A malicious entity, called the *Attacker*, is explicitly represented in the system. The rest of the entities in the system may inadvertently help the Attacker by *leaking* their sensitive data. Equipped with the acquired knowledge the Attacker can then intrude the victim entities. Therefore, the system's vulnerabilities are also explicit in the model, and are represented by leaks.

We also propose a correct-by-construction transformation of an IoT model into a stochastic component-based model, called $\mathcal{S}BIP$ [2, 4], for which an execution engine is developed and maintained. We can therefore execute our IoT model and run several verification and analysis tests, such as deadlock detection. Moreover, the attack tree provided with the model is transformed into a *monitor* that observes the interactions the Attacker has with the system and analyse when an attack is successful.

We then ask what is the probability of a successful attack given an IoT system and an attack tree. To respond to this question, we use two methods of statistical model checking (SMC) [4]: *Monte Carlo*, a standard SMC method, and *importance splitting* [9]. The Monte Carlo method consists of sampling the executions of an IoT system and computing the probability of an attack based on the number of executions for which the attack was successful. A successful attack can be considered a rare event if its probability value is in the range of 10^{-5} or 10^{-6}. For rare events, the Monte Carlo method can be problematic as it requires a large number of simulations for a correct estimate. We therefore use a second SMC method, developed for rare events, called *importance splitting* [9]. Importance splitting assumes that an execution leading to a rare event can be decomposed into several intermediate steps. Instead of executing a system until the rare event occurs, the execution is stopped after one of the intermediate steps is reached. The execution is restarted then from that step onward. Not only importance splitting can infer the probability of a rare event but it is also well suited for attack trees. The intermediate steps leading to a rare event are deduced from the nodes in the tree leading to a successful attack.

We implemented a tool chain to automate the analysis presented above. It consists of a compiler from our IoT modeling language into $\mathcal{S}$BIP. The execution engine of $\mathcal{S}$BIP is then used to produce simulations of the system, which are then fed to Plasma [5], a model checker that implements both Monte Carlo and importance splitting. Throughout our paper, we use a running example involving cyber-attacks on a Smart Hospital. The example is based on existing attacks carried against hospitals IT system as reported by TrapX [1] and ENISA [6].

The paper is structured as follows. Section 2 presents the IoT modeling language, Section 3 introduces attack trees and Section 4 presents $\mathcal{S}$BIP. The transformation from IoT to $\mathcal{S}$BIP is shown in Section 5. Section 6 presents the two SMC techniques. In Section 7 we validate our approach using some experiments on the running example. Related works that we are aware of, are summarised and compared with our work in section 8. Lastly, Section 9 concludes.

2 Probabilistic IoT Models

The components of an IoT system, called *entities*, have each a *knowledge*, used to allow (or disallow) its interaction with the rest of the system. For instance, an entity can send an email to another entity only if it *knows* its email address. Or an user needs to *know* the `url` of a website in order to access it; we say that the `url` is part of the user's knowledge. For simplicity, we represent knowledge as a finite set of *values*.

Protocols are used at each interaction to verify the knowledge of the interacting entities. Each value is associated to a protocol. Two entities can communicate through a protocol if they have a common value for that protocol. We write C for a set of protocols, ranged over by c and Val for a set of values ranged over by v.

2.1 Processes and States

Each entity has a unique identifier, denoted by $e_1, \cdots e_n$ and a running process. The grammar of processes is defined in Figure 1.

$$\begin{array}{rrl}
\textit{Process} & P, Q & ::= T \;||\; P \mid Q \\
\textit{Thread} & T, U & ::= 0 \;||\; A \;||\; \sum_{i \in I} [n_i] a_i . T_i \text{ where } n_i \in (0,1] \text{ and } \sum_{i \in I} n_i = 1 \\
\textit{Action} & a, b & ::= e \xrightarrow[\mathrm{v}]{\mathrm{c}} e' \quad (\textsc{Send}) \;||\; e \xleftarrow{\mathrm{c}} e' \quad (\textsc{Receive}) \;|| \\
 & & \quad e \xrightarrow[\mathrm{v}]{\quad}\!\!\!\!\twoheadrightarrow e' \quad (\textsc{Leak}) \;||\; e \twoheadleftarrow e' \quad (\textsc{Collect}) \;|| \\
 & & \quad \tau \quad (\textsc{Internal}) \\
\textit{Definition} & A & \overset{\text{def}}{=} T \\
\textit{State} & s & ::= \emptyset \;||\; \langle P, k \rangle \;||\; s \mid s.
\end{array}$$

Fig. 1: Syntax of the probabilistic IoT-calculus

Processes are composed of *threads*, using the parallel composition operator. A thread can only do sequential computations. We write 0 for the inactive thread and A for the (recursive) definitions of threads.

The actions of a thread consists of sending and receiving values under an agreed upon protocol. We distinguish between "safe" interactions and the ones that can potentially lead to security issues, called *leak* and *collect*. A *leak* is a send action where there is no protocol governing the interaction and *collect* is its receive counterpart. Processes can also do silent moves, denoted by τ. Moreover, actions are equipped with a probability, denoted by $[n]a$, for an action a and a probability $n \in [0, 1]$. Threads can therefore do a probabilistic choice between actions, with the restriction that the sum of the probabilities of all available actions is 1. If there is only one available action, its probability is 1 and can be omitted.

A *knowledge* function $K : E \times C \to \mathcal{P}(\mathit{Val})$ associates a set of values to each entity and protocol. For simplicity we write k_i^c for the knowledge of entity i under protocol c. The function $\mathsf{protocol} : \mathit{Val} \to C$ associates each value to a protocol.

Each entity, has at any state of its computation, a running process P and a knowledge k. The grammar for states is included in Figure 1. The global state of a IoT system consists of the parallel composition of all entities states $s_1 \mid \cdots \mid s_n$, where s_i is the current state of the entity e_i.

2.2 Operational semantics

We define $\equiv_P \subseteq P \times P$ to be the smallest congruence on processes which includes the associativity and the commutativity for $+$ and $|$; the identity element 0 for $|$ and the unfolding law for definitions: $A \equiv_P T$ if $A \stackrel{\text{def}}{=} T$. We also introduce a congruence relation on states $\equiv_s \subseteq s \times s$ which includes the associativity and the commutativity for $|$, the identity element $\emptyset$ and which generalizes the congruence on processes: if $P \equiv_P Q$ then $\langle P, k\rangle \equiv_s \langle Q, k\rangle$.

The operational semantics of Figure 2, defines a transition system (S, T, L, s_0) where we write S for the set of states, with s_0 the initial state, $L \subseteq \{\tau\} \cup (\{\text{SR, LC}\} \times \mathit{Val})$ for a set of labels, ranged over by l, and $T \subseteq S \times [0, 1] \times L \times S$ for a set of transitions, where each transition is decorated by a probability and by a label. A transition can either be internal, labeled by τ, or it can be an interaction between two entities exchanging a value.

In our semantics, a probabilistic choice is always resolved locally, using the Choice rule. A transition derived by the Choice rule is considered internal and is labeled τ. A process can also do internal transitions using rule Internal. Rule SendReceive defines the interaction between two components e_1 and e_2. The interaction is allowed if the sender and the receiver share some common values under the protocol c. After the interaction, the receiver's knowledge is updated by adding the received value under the corresponding protocol. A LeakCollect interaction proceeds similarly, except that there are no checks on the knowledge of the two components. Rules Congruence and ParProc allows one to use congruence and parallel composition on states to derive transitions.

Choice

$$\langle \sum_{i \in I} [n_i] a_i.T_i, k\rangle \xrightarrow[\tau]{[n_i]} \langle a_i.T_i, k\rangle$$

Internal

$$\langle \tau.P, k\rangle \xrightarrow[\tau]{[1]} \langle P, k\rangle$$

SendReceive

$$\frac{\exists v \in k_1^c \text{ s.t. } v \in k_2^c \qquad c' = \mathsf{protocol}(v')}{\langle e_1 \xrightarrow[\text{v'}]{\text{c}} e_2.P_1, k_1\rangle | \langle e_2 \xleftarrow{\text{c}} e_1.P_2, k_2\rangle \xrightarrow[\text{SR}:v']{[1]} \langle P_1, k_1\rangle | \langle P_2, k_2^{c'} \uplus \{v'\}\rangle}$$

LeakCollect

$$\frac{c' = \mathsf{protocol}(v')}{\langle e_1 \xrightarrow[\text{v'}]{} \!\!\!\!\!\rightarrow e_2.P_1, k_1\rangle | \langle e_2 \twoheadleftarrow e_1.P_2, k_2\rangle \xrightarrow[\text{LC}:v']{[1]} \langle P_1, k_1\rangle | \langle P_2, k_2^{c'} \uplus \{v'\}\rangle}$$

ParProc

$$\frac{\langle P_i, k_i\rangle | \langle P_j, k_j\rangle \xrightarrow[l]{[n]} \langle P_i', k_i'\rangle | \langle P_j', k_j'\rangle}{\langle P_i \mid Q_i, k_i\rangle | \langle P_j \mid Q_j, k_j\rangle \xrightarrow[l]{[n]} \langle P_i' \mid Q_i, k_i'\rangle | \langle P_j' \mid Q_j, k_j'\rangle}$$

Congruence

$$\frac{s \equiv_s t \xrightarrow[l]{[n]} s' \equiv_s t'}{s \xrightarrow[l]{[n]} s'}$$

ParState_Tau

$$\frac{\langle P, k\rangle \xrightarrow[\tau]{[n]} \langle P', k'\rangle \qquad \mathsf{count}_\tau(\langle P, k\rangle | s) = m}{\langle P, k\rangle | s \xrightarrow[\tau]{[1/m \cdot n]} \langle P', k'\rangle | s}$$

ParState_Interaction

$$\frac{\langle P_i, k_i\rangle | \langle P_j, k_j\rangle \xrightarrow[l]{[1]} \langle P_i', k_i'\rangle | \langle P_j', k_j'\rangle \qquad \mathsf{count}_{\mathtt{SR,LC}}(\langle P_i, k_i\rangle | \langle P_j, k_j\rangle | s) = m \quad \mathsf{count}_\tau(\langle P_i, k_i\rangle | \langle P_j, k_j\rangle | s) = 0}{\langle P_i, k_i\rangle | \langle P_j, k_j\rangle | s \xrightarrow[l]{[1/m]} \langle P_i', k_i'\rangle | \langle P_j', k_j'\rangle | s}$$

Fig. 2: The operational semantics of an IoT system

The rules for the global states, ParState_Tau and ParState_Interaction, give priority to the internal transitions over the binary interactions. Moreover, in each case, we choose a global transition from several local ones using an uniform distribution. We rely on two auxiliary functions, count_τ and $\mathsf{count}_{\mathtt{SR,LC}}$ that count the number of local transitions with labels τ and labels SR, LC, respectively.

Definition 1. *Given an IoT model* (S, T, L, s_0) *with an initial state* s_0*, an* execution *is a sequence of transitions in* T, $\sigma = \{s_i \xrightarrow[l_i]{[n_i]} s_i'\}_{0 \le i \le k}$*, such that* $s_0 = s$ *and* $\forall i \ge 1$, $s_{i-1}' = s_i$*. The probability of a transition* $s \xrightarrow[l]{[n]} s'$ *is* n *and* the probability of an execution σ *is* $\prod_{0 \le i \le k} n_i$.

Example 1. Let us now introduce our running example, the Smart Hospital system. Let $E = \{A(ttacker), H(ospital), E(mployee)\}$ be three entities which communicate with each other using the protocols $C = \{url, message, mail, phone\}$. We introduce the following actions:

$$\text{AH} = A \xrightarrow[\text{getSensitiveData}]{\text{url}} H \qquad \text{leakEmail} = H \xrightarrow[\text{emailEmployee}]{} \!\!\!\!\!\rightarrow A$$

$$\begin{aligned}
\text{HA} =& H \xhookleftarrow{\text{url}} A & \text{leakPhone} =& H \xrightarrow[\text{phoneEmployee}]{\hspace{4em}}\!\!\!\!\rightarrow A \\
\text{AE\_mail} =& A \xrightarrow[\text{getCredential}]{\text{mail}} E & \text{leakIssue} =& H \xrightarrow[\text{issueEmployee}]{\hspace{4em}}\!\!\!\!\rightarrow A \\
\text{EA\_mail} =& E \xhookleftarrow{\text{mail}} A & \text{leakCredentials} =& E \xrightarrow[\text{credEmployee}]{\hspace{4em}}\!\!\!\!\rightarrow A \\
\text{AE\_phone} =& A \xrightarrow[\text{getCredential}]{\text{phone}} E & & \\
\text{EA\_phone} =& E \xhookleftarrow{\text{phone}} A & &
\end{aligned}$$

and process definitions:

$$\begin{aligned}
\text{Attacker} =& \text{AChoice} \mid \text{collectH} \mid \text{collectE} \\
\text{AChoice} =& [0.4]\text{AH.AChoice} + [0.3]\text{AE\_mail.AChoice} + [0.3]\text{AE\_phone.AChoice} \\
\text{collectH} =& A \twoheadleftarrow H.\text{collectH} \\
\text{collectE} =& A \twoheadleftarrow E.\text{collectE} \\
\text{Hospital} =& \text{HA.}([n_1]\text{leakPhone.Hospital} + [n_2]\text{leakIssue.Hospital}+ \\
& [n_3]\text{leakEmail.Hospital} + [n_4]\tau.\text{Hospital}) \\
\text{Employee} =& [m_1]\text{EA\_mail.EChoice} + [m_2]\text{EA\_phone.EChoice} \\
\text{EChoice} =& [m_3]\text{leakCredentials.Employee} + [m_4]\tau.\text{Employee}
\end{aligned}$$

A has Attacker as initial process and similarly for H and E. Their initial knowledge is:

$$\begin{aligned}
k_A =& \{\mathit{url} = \{\text{urlHospital}\}, \mathit{message} = \{\text{getSensitiveData, getCredentials}\}\} \\
k_H =& \{\mathit{url} = \{\text{urlHospital}\}, \mathit{mail} = \{\text{emailEmployee}\}, \mathit{phone} = \{\text{phoneEmployee}\}\} \\
k_E =& \{\mathit{mail} = \{\text{emailEmployee}\}, \mathit{phone} = \{\text{phoneEmployee}\}\}
\end{aligned}$$

where the missing protocols are initially the emptysets.

Let us now consider the transitions below. The Attacker starts by choosing to contact the Hospital using the internal transition (1), after which the two entities can communicate in transition (2). At this point the Hospital can either leak some sensitive information (*emailEmployee*, *phoneEmployee* or *issueEmployee*) or it can do an internal transition. Transitions $(3) - (4)$ represent the scenario where *emailEmployee* is leaked. The knowledge of the attacker is augmented with the leaked data: $k_A^{mail} \cup \{\text{emailEmployee}\}$ and the attacker can now communicate with the employee using the email protocol.

$$\langle \text{Attacker}, k_A \rangle \mid \langle \text{Hospital}, k_H \rangle \xrightarrow[\tau]{[0.4]} \tag{1}$$

$$\langle \text{AH.AChoice} \mid \text{collectH} \mid \text{collectE}, k_A \rangle \mid \langle \text{Hospital}, k_H \rangle \xrightarrow[\text{SR:getSensitiveData}]{[1]} \tag{2}$$

$$\langle \text{Attacker}, k_A \rangle \mid \langle [n_3]\text{leakEmail.Hospital} + \cdots, k'_H \rangle \xrightarrow[\tau]{[n_3]} \tag{3}$$

$$\langle \text{Attacker}, k_A \rangle \mid \langle \text{leakEmail.Hospital}, k'_H \rangle \xrightarrow[\text{LC:emailEmployee}]{[1]} \tag{4}$$

$$\langle \text{Attacker}, k'_A \rangle \mid \langle \text{Hospital}, k'_H \rangle$$

Suppose that after transition (2), the Hospital does not leak any information:

$$\langle \text{Attacker}, k_A \rangle \mid \langle [n_3]\text{leakEmail.Hospital} + \cdots, k'_H \rangle \xrightarrow[\tau]{[n_4]}$$
$$\langle \text{Attacker}, k_A \rangle \mid \langle \text{Hospital}, k'_H \rangle.$$

Then the Attacker cannot communicate with the Employee. If the Attacker tries to communicate without knowing the *emailEmployee*, the system deadlocks. A sequence of transitions ending with a deadlock represents an unsuccessful attack.

3 Attack Trees

In this section we formally introduce attack trees and we show how attack trees can be used to monitor the execution of an IoT system.

Figure 3 shows an attack tree for the Smart Hospital in Example 1. The root of the tree is the main goal of the Attacker, which is getting the Employee's credentials. The nodes represent the possible attacks. For example the *qui pro quo attack* consists of contacting the Employee and posing as a technician. For this attack, node *get information* has to happen as well. It requires for the Attacker to get the Employee's phone number and technical issue, which both are leaked by the Hospital. The second possibility is a *phishing attack*. It consists of the Attacker contacting the Hospital and then the Hospital leaking the Employee's email. If either of the two attacks succeed, the Attacker can then try to get the Employee's credentials.

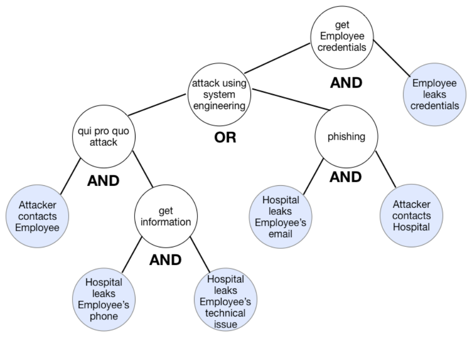

get Employee credentials
AND
attack using system engineering
OR
Employee leaks credentials
qui pro quo attack
AND
phishing
AND
Attacker contacts Employee
get information
AND
Hospital leaks Employee's email
Attacker contacts Hospital
Hospital leaks Employee's phone
Hospital leaks Employee's technical issue

Fig. 3: An attack tree for the Smart Hospital

Formally, the leaves of the tree corresponds to *events* happening in the system. An event in an IoT system consists of the exchange of a value between some entities. For example, the node *Employee leaks credentials* stands for the pair *(LC, "credEmployee")*, meaning that the value *credEmployee* has been leaked, through a LeakCollect interaction.

Definition 2. *Let $\varDelta \subseteq \{SR,LC\} \times Val$ be a set of events. An* attack tree $\mathtt{t}$ *is a term constructed recursively from the set $\varDelta$ using the operators $\vee$ and $\wedge$.*

An attack tree can also be seen as a Boolean expression, by associating to every event $e \in \varDelta$ a Boolean variable v_e.

Definition 3. *Let t be an attack tree. The* semantics of t, *denoted $[\![t]\!]$, consists of a Boolean expression defined by recursion on t as follows:*

- *if $e \in \varDelta$ then $[\![t]\!] = v_e$;*
- *if $t = t_1 \wedge t_2$ then $[\![t]\!] = [\![t_1]\!] \wedge [\![t_2]\!]$;*
- *if $t = t_1 \vee t_2$ then $[\![t]\!] = [\![t_1]\!] \vee [\![t_2]\!]$.*

Let $\mathbf{X} : \varDelta \rightarrow \{true, false\}$ be a valuation for $\varDelta$, then the semantics of $\mathtt{t}$ w.r.t. $\mathbf{X}$, denoted $[\![\mathtt{t}]\!](\mathbf{X}) \in \{true, false\}$, consists in evaluating the associated Boolean formula [10].

In order to assess whether an attack was successful, we can use an attack tree to monitor the executions of an IoT system. Given an execution trace σ, its corresponding valuation $\mathbf{X}(\sigma)$ sets v_e to *true* if the event e occurred in σ. If $[\![t]\!](\mathbf{X}(\sigma))$ is true then the execution σ is a successful attack of $\mathtt{t}$.

4 SBIP: A Stochastic Component Based Model

$\mathcal{S}$BIP [4, 2] is a stochastic, component based framework that allows modeling hierarchical systems from *atomic* components. We introduce $\mathcal{S}$BIP in four steps: we start with some preliminary notations; then we introduce the syntax of the atomic components; next its semantics; and lastly we explain how to compose atomic components into hierarchical systems.

4.1 Preliminaries

Let V be a set of variables, and for each variable $v_j \in V$, let D_j be its data domain, denoted by $v_j : D_j$. A *valuation* for the variables in V is a function $\mathbf{X} : V \rightarrow \cup_j D_j$ which assigns values to variables. We denote $\mathbf{X}(v)$ the valuation of the variable $v \in V$.

Let $\mathbb{E}$ be a set of operators. We denote by $\mathbb{E}[V]$ the set of expressions constructed from a set of variables V and operators. A function $f(V)$ is then just an expression in $\mathbb{E}[V]$. We denote $\mathbf{X}(e)$ the valuation of the expression $e \in \mathbb{E}[V]$ according to the valuation of the variables in V.

We write $v := e$ for an *assignment*, or *update* of v, and write $\mathbb{A}[V]$ for a set of assignments for the variables in V.

We distinguish between two types of variables: the deterministic variables and the *random* variables, used for encoding the stochastic behavior. A random variable v is associated with a probability distribution μ over its valuation domain D, denoted as $v \sim \mu$, where $\mu : D \rightarrow [0, 1]$ and $\sum_{x \in D} \mu(x) = 1$.

Lastly, we denote $(f \circ g)(x) = f(g(x))$ the composition of functions. If the two functions have disjoint domains, we write $f \sqcup g$ for their disjoint composition. We use $\cup$ for set union and $\uplus$ for disjoint union.

4.2 Stochastic Atomic Components

$\mathcal{S}$BIP components are 1-safe Petri-Nets equipped with (i) ports that allow the component to communicate with other components; and (ii) variables that can be read and updated during communications.

Definition 4. *A* stochastic atomic component *consists of the tuple $\mathcal{B} = (P, V, N)$, where*

- *P is a set of communication ports.*
- *$V = V^d \uplus V^p$, with $V^d = \{v_1, \ldots, v_n\}$ a set of deterministic variables and $V^p = \{v_1^p, \ldots, v_m^p\}$ a set of random variables with an associated distribution $v_i^p \sim \mu_i$.*
- *$N = (L, L_0, T)$ is a Petri-Net[4] where L is a set of places and $L_0 \subseteq L$ are the initial places. T is a set of transitions $t = ({}^\bullet t, \langle p, g, f \rangle, t^\bullet)$ where ${}^\bullet t$ (resp. $t^\bullet$) is the set of input (resp. output) places of t. Transitions are labeled by the triple $\langle p, g, f \rangle$ where $p \in P$ is a port, $g \in \mathbb{E}[V]$ is a* guard *and $f = (f^d, R^p)$ is an* update *function, such that $f^d = \{v := f(V) \mid v \in V^d\} \in \mathbb{A}[V]$ is a set of functions that update the deterministic variables and $R^p \subseteq V^p$ is a subset of random variables to be updated.*

We sometimes write p_t, g_t and f_t^d, R_t^p for the label of t. We define *markings* as the set of functions $m : L \to \{0, 1\}$. Given two markings m_1, m_2 we define inclusion $m_1 \leq m_2$ iff for all $l \in L$, $m_1(l) \leq m_2(l)$. Also, we define addition $m_1 + m_2$ as the marking m_{12} such that, for all $l \in L$, $m_{12}(l) = m_1(l) + m_2(l)$.

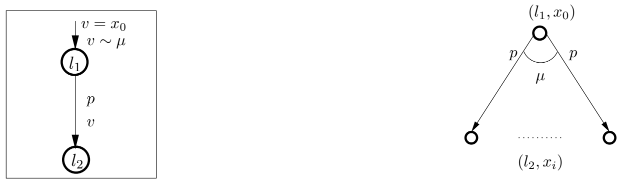
A *priority order* on a set of ports is a partial order, where each element $p < p'$ of the order is called a *priority*. Whenever the system has a choice between the two interactions on two ports p or p', the interaction on p' is chosen.

4.3 Semantics of Stochastic Atomic Components

The semantics of a $\mathcal{S}$BIP component $\mathcal{B} = (P, V, N)$ consists of a transition system $\mathcal{M}$, where the states are of the form $(m, \mathbf{X})$, for m a marking of N and $\mathbf{X}$ a valuation of V.

The random variables engender a probabilistic behavior over transitions of $\mathcal{M}$. Let us consider an atomic component $\mathcal{B}$ in Figure 4a that has a transition going from place l_1 to place l_2 using port p, with a guard that is always true, and which updates a random variable v with the valuation domain D and distribution μ. Assuming the initial value of v is x_0, when executing $\mathcal{B}$, there will be several possible transitions, shown in Figure 4b, from state $(\{l_1\}, x_0)$ to states $(\{l_2\}, x_i)$ for all $x_i \in D$. The probabilities of these transitions is given by μ. Since the random variables are independent, when several random variables are updated, the resulting distribution on transitions is the product of the distributions associated to each variable.

[4] N is equivalent to the extended 1-safe Petri-Net (L, L_0, T, F) where $F = \{(l, t) \mid l \in {}^\bullet t\} \cup \{(t, l) \mid l \in t^\bullet\}$ is the token flow relation and can be deduced from T.



(a) A Stochastic Atomic Component $\mathcal{B}$ (b) Behavior of $\mathcal{B}$

Fig. 4: Example of a stochastic atomic component $\mathcal{B}$ and its behavior.

Atomic components with random variables lead to behaviors that combine both stochastic and non-deterministic aspects. A transition is possible if a communication is ready on its associated port. At any given state, several ports can be ready for a communication, and thus several transitions can be enabled, regardless of whether they are associated or not with random variables. Non-determinism is always resolved in $\mathcal{S}$BIP to a probabilistic choice on an uniform distribution. To formally state this, we denote with $\mathsf{Enabled}(m; \mathbf{X})$ the set of transitions in T that are enabled by m for a valuation $\mathbf{X}$: $\mathsf{Enabled}(m; \mathbf{X}) = \{t \in T \mid {}^\bullet t \leq m$ and $\mathbf{X}(g_t)$ is true$\}$[5].

Definition 5. *The* semantics of a stochastic component $\mathcal{B} = (P, V, (L, L_0, T))$ *with* $\mathbf{X_{init}}$ *an initial valuation, is defined as a probabilistic transition system* $\mathcal{M} = \langle Q, \pi, P, q_0 \rangle$*, where:*

- *Q is a set of states of the form $(m, \mathbf{X})$; $q_0 = (m_0, \mathbf{X_{init}})$ is the initial state where m_0 is the marking associated to L_0, i.e. $m_0(l) = 1 \iff l \in L_0$ and 0 otherwise;*
- *$\pi \subseteq Q \times P \times Q$ is a set of transitions defined by the following rule:*

$$\frac{\begin{array}{c} t \in T \quad {}^\bullet t \leq m \quad m' = m - {}^\bullet t + t^\bullet \quad \mathbf{X}(g_t) = \mathit{true} \\ \mathbf{X}' = [v^d := \mathbf{X}(f_t^d), v^p := \mathit{random}(\mu)] \quad v^d \in V^d \quad v^p \in R_t^p, v^p \sim \mu \end{array}}{(m, \mathbf{X}) \xrightarrow{p_t} (m', \mathbf{X}')}$$

Lastly, we defined the probability of a transition as follows:

$$\mathbb{P}\big(q \xrightarrow{p} q'\big) = \frac{1}{|\mathsf{Enabled}(m; \mathbf{X})|} \cdot \prod_{v_i \in R^p, v_i \sim \mu_i} \mu_i(X'(v_i)).$$

In the definition above we say that the state $(m', \mathbf{X}')$ is a successor of state $(m, \mathbf{X})$, if t is a transition of T enabled by the marking m, the guard g_t evaluates to *true* and the new valuation $\mathbf{X}'$ on the variables $V^d \cup V^p$ is obtained by applying f_t^d on the deterministic variables V^d and updating the random variables in R_t^p. The probability of a transition is computed by first selecting a transition with an uniform distribution from the set of enabled transitions; and then, selecting the next state according to the distributions attached to the random variables.

[5] Remark that the cardinality of $\mathsf{Enabled}(m; X)$ can be greater than one.

4.4 Composition of Stochastic Components

Definition 6. *An* interaction $\gamma = (P, G, F)$ *on a set of components* $\mathcal{B}_i = (P_i, V_i, N_i)$*, for* $i \in I$*, where* I *is set of indexes, consists of:*

- $P = \{p_i \mid p_i \in P_i, i \in I\}$ *is a disjoint set of ports containing exactly one port from each* $\mathcal{B}_i$*,* $i \in I$*;*
- G *is a global guard defined on* $V_\gamma = \cup_{i \in I} V_i$*;*
- $F = \{v := F(V_\gamma) \mid v \in \cup_{i \in I} V_i^d\}$ *is a global update function used to exchange values between components.*

For n atomic components and for Γ a set of interactions, we write $\Gamma(\mathcal{B}_1, \ldots, \mathcal{B}_n)$ their composition into a stochastic component. Intuitively the local transitions of the atomic components synchronise to produce global transitions using the interactions.

Definition 7. *Let* Γ *be a set of interactions defined on* n *components* $\mathcal{B}_i = (P_i, V_i, N_i)$*, with* $N_i = (L_i, L_{0,i}, T_i)$ *for* $i \leq n$*. The composition of the* n *components, denoted as* $\Gamma(\mathcal{B}_1, \ldots, \mathcal{B}_n)$*, is a stochastic component* $\mathcal{B} = (\Gamma, V, N)$*, with* $N = (L, L_0, T)$*, defined as follows:*

- $V = \cup_{i \leq n} V_i$*;*
- $L = \cup_{i \leq n} L_i$ *with* $L_0 = \cup_{i \leq n} L_{0,i}$ *;*
- $T = \left\{ ({}^\bullet T_\gamma, \langle \gamma, g, f \rangle, T_\gamma^\bullet) \mid \gamma \in \Gamma \right\}$*, where* $T_\gamma = \{t_i \mid p_i \in P_\gamma\}$ *is the set of transitions that synchronize on the interaction* $\gamma \in \Gamma$*. Then* ${}^\bullet T_\gamma = \{l \mid l \in {}^\bullet t_i, t_i \in T_\gamma\}$ *and* $T_\gamma^\bullet = \{l \mid l \in t_i^\bullet, t_i \in T_\gamma\}$*. Each transition is labeled by the triple* $\langle \gamma, g, f \rangle$ *where* $g = G_\gamma \wedge (\bigwedge_{t_i \in T_\gamma} g_{t_i})$ *and* $f = (\sqcup_{t_i \in T_\gamma} f_{t_i}) \circ F_\gamma$ *consists of the composition of all* f_{t_i} *with* F_γ*.*

Assembling stochastic atomic components produces a stochastic atomic component, and thus its semantics is given by Definition 5.

We use a priority order, denoted $\ll$, which gives priority to the internal transitions over the binary interactions. We write then $\langle \ll \rangle(\Gamma(\mathcal{B}_1, \ldots, \mathcal{B}_n))$ for a $\mathcal{S}$BIP system.

5 Transformation from IoT to $\mathcal{S}$BIP

We now show how to transform an IoT system to $\mathcal{S}$BIP. Entities of an IoT model become atomic components and their communications is represented as interactions.

An entity can have several threads running, all sharing the same knowledge. To model this we encode each thread of a process into a Petri Net (Definition 8). The encoding of a process is then the union of the several Petri Nets, which all have a common set of variables, guards and update functions (Definition 9).

The deterministic variables are used to model the entity's knowledge. The random variables, similarly to [4], encode the probabilities associated to actions in a summation process.

We use labeling functions on places and on the random variables, denoted by ℓ. The labels are the threads of the original IoT system. Moreover we identify places that have congruent labels, i.e. $l_1 \equiv_L l_2 \iff \ell(l_1) \equiv_P \ell(l_2)$. We write l_T and v_T when $\ell(l) = T$ and $\ell(v) = T$, respectively.

Definition 8. *For a thread T, let* Definitions *and* Actions *be the sets of thread definitions and of actions, respectively, used recursively in T. We define the* transformation of T *to be the atomic component* $(\mathsf{Actions}, V^d \uplus V^p, (\mathcal{L}, \mathcal{L}_0, \mathcal{T}))$ *with:*

- $V^d = \{v_c \mid c \textit{ is a protocol used in } T\}$;
- $V^p = [\![T]\!]_v \cup \{[\![U]\!]_v \mid A \overset{def}{=} U \textit{ and } A \in \mathsf{Definitions}\}$;
- $\mathcal{L} = \left([\![T]\!]_s \cup \{[\![U]\!]_s \mid A \overset{def}{=} U \textit{ and } A \in \mathsf{Definitions}\}\right)_{\equiv_L}$ *is a set of places partitioned in equivalence classes by the $\equiv_L$ relation, with $\mathcal{L}_0 = \{l_T\}$;*
- $\mathcal{T} = [\![T]\!]_t \cup \{[\![U]\!]_t \mid A \overset{def}{=} U \textit{ and } A \in \mathsf{Definitions}\}$.

In our transformation we use the functions $[\![\cdot]\!]_v$, $[\![\cdot]\!]_s$ and $[\![\cdot]\!]_t$ to transform a thread into a set of places, random variables and transitions. The functions are formally defined in Figure 5. Intuitively, for each possible continuation of T we introduce a new place, and we use the labeling function on places to keep track of the correspondence between places and threads. Also whenever T is of the form $\sum_{i \in I}[n_i]a_i.T_i$ we introduce a new place, denoted $l^\star_T$. This additional place is where the choice between the different branches of the sum is made. The random variables are defined using the function $[\![\cdot]\!]_v$. Whenever T is of the form $\sum_{i \in I}[n_i]a_i.T_i$ we introduce a new random variable v_T. The valuation domain D for v_T is the set of states associated to the possible continuations i.e. $D = \{a_i.T_i\}_{i \in I}$. The probability distribution of v_T is defined by the probabilities n_i i.e. $\mu(a_i.T_i) = n_i$. Lastly, $[\![T]\!]_s$ defines the transitions. The guards are only used when making a probabilistic choice: Suppose we are currently running thread T and we wish to go from state $l^\star_T$ to a state l_{T_i}. The guard then checks that the value of the random variable v_T is updated to $a_i.T_i$. For the rest of transitions, the guard is the constant *true*.

Definition 9. *Let e be an entity in an IoT system with the initial state $s_0 = \langle P, k\rangle$ and $P = T_1 \mid \cdots \mid T_m$. Let $(P^j, V^j, (\mathcal{L}^j, \mathcal{L}^j_0, \mathcal{T}^j))$ be the atomic components obtained from each T_j, $j \le m$.*

We define the transformation of e as the atomic component $\mathcal{B}_e = (P, V, N)$ *with $P = \cup_{j \le m} P_j$, $V^d = \cup_{j \le m} V^d_j$, $V^p = \uplus_{j \le m} V^p_j$ and with $N = (\uplus_{j \le m}\mathcal{L}^j$, $\uplus_{j \le m}\mathcal{L}^j_0, \uplus_{j \le m}\mathcal{T}^j)$. We also define the initial valuation $\mathbf{X_{init}}(v_c) = k(c)$ where for each protocol c we initialize the variable v_c to the set of values $k(c)$.*

Note that we are using set union for ports and variables as the different threads of an entity share their ports and knowledge. However, in order to clearly separate the behaviour of the different threads, we use disjoint union when combining the Petri Nets of the different threads. This is allowed because the different threads in an entity cannot interact with each other, but only with other entities.

$$
\begin{aligned}
[\![\sum_{i\in I}[n_i]a_i.T_i]\!]_v =& \bigcup_{i\in I}[\![a_i.T_i]\!]_v \cup \{v_T \mid v_T \sim \mu \text{ s.t. } \mu(a_i.T_i) = n_i, \forall i \in I\}\\
& \text{where } T = \sum_{i\in I}[n_i]a_i.T_i \text{ and } |I| > 1\\
[\![a.T]\!]_v =& [\![T]\!]_v\\
[\![A]\!]_v = [\![0]\!]_v =& \emptyset\\
[\![\sum_{i\in I}[n_i]a_i.T_i]\!]_s =& \bigcup_{i\in I}[\![T_i]\!]_s \cup \{l_T, l_T^\star\},\ \text{where } T = \sum_{i\in I}[n_i]a_i.T_i \text{ and } |I| > 1\\
[\![a.T]\!]_s =& [\![T]\!]_s \cup \{l_{a.T}\}\\
[\![A]\!]_s =& \{l_A\}\\
[\![0]\!]_s =& \{l_0\}\\
[\![\sum_{i\in I}[n_i]a_i.T_i]\!]_t =& \bigcup_{i\in I}\Big((\{l_T^\star\}, \langle a_i, g = (v_T == a_i.T_i), f\rangle, \{l_{T_i}\}) \cup [\![T_i]\!]_t\Big)\\
& \cup (\{l_T\}, \langle \tau, \text{true}, f^\star\rangle, \{l_T^\star\}) \text{ where } T = \sum_{i\in I}[n_i]a_i.T_i \text{ and } |I| > 1\\
[\![a.T]\!]_t =& (\{l_{a.T}\}, \langle a, \text{true}, f\rangle, \{l_T\}) \cup [\![T]\!]_t\\
[\![0]\!]_t = [\![A]\!]_t =& \emptyset
\end{aligned}
$$

where $f = \{v := v \mid v \in V^d\}$ and $R^p = \emptyset$ and $f^\star$ defined as f but with $R^p = \{v_T\}$.

Fig. 5: The functions used in the transformation in Definition 8

Communications between two entities e_1 and e_2 in the IoT language are transformed into a set of guarded interactions between components $\mathcal{B}_{e_1}$ and $\mathcal{B}_{e_2}$.

Definition 10. *Let $\mathcal{B}_{e_i} = (P_i, V_i, N_i)$ be the transformation of an IoT system with n entities e_i and with the initial state s_0. For all $a \in \mathsf{Actions}$, if there exists $a' \in \mathsf{Actions}$ such that*

- *either $a = e_1 \underset{v}{\overset{c}{\rightarrow}} e_2$ and $a' = e_2 \overset{c}{\leftarrow} e_1$,*
- *or $a = e_1 \underset{v}{\longrightarrow\!\!\!\!\rightarrow} e_2$ and $a' = e_2 \twoheadleftarrow e_1$*

then we define an interaction γ=$(\{a, a'\}, G, F)$ where

- *if $a = e_1 \underset{v}{\overset{c}{\rightarrow}} e_2$ then $G = (\exists x \in v_c^1$ such that $x \in v_c^2)$ for $v_c^1 \in V_1^d$, $v_c^2 \in V_2^d$; otherwise $G = true$;*
- *$F = \{v_{c'}^2 := v_{c'}^2 \cup \{v\} \mid \mathsf{protocol}(v) = c', v_{c'}^2 \in V_2^d\}$*

and where V_1^d, V_2^d are the deterministic variables of $\mathcal{B}_{e_1}$ and $\mathcal{B}_{e_2}$, respectively.

We also define the interaction $(\{\tau\}, true, F)$, where $F = \{v := v \mid v \in V^d\}$, for every component $\mathcal{B}_e = (P, V, N)$.

Note that interactions are only defined for consistent pairs of *SendReceive* and *LeakCollect* reflecting the rules in Figure 2.

Given an IoT system with n entities and an initial state s_0 let us write Γ for the set of interactions of Definition 10. Recall that $\ll$ is the priority order of Section 4.4 and that $\Gamma(\mathcal{B}_1, \ldots, \mathcal{B}_n)$ is the composition of the n entities, as in Definition 7. Then $\langle\ll\rangle(\Gamma(\mathcal{B}_1, \ldots, \mathcal{B}_n))$ is the resulting $\mathcal{S}$BIP system. We have everything in place to show a correspondence between the *semantics* of the IoT system and of its $\mathcal{S}$BIP transformation.

Theorem 1. *Let (S, L, T, s_0) be an IoT system where $s_0 = \langle P_1, k_1\rangle \mid \cdots \langle P_n, k_n\rangle$ and where $\langle P_i, k_i\rangle$ is the initial state of each enity e_i, $i \leq n$. Let $\mathcal{B}_{e_i}$ be the $\mathcal{S}$BIP transformation and $\mathbf{X}^{\mathbf{i}}_{\mathbf{init}}$ be the initial valuation of the entity e_i and let Γ be the corresponding set of interactions. Lastly, let $\mathcal{M} = (Q, \pi, P, q_0)$ be the semantics of $\langle\ll\rangle(\Gamma(\mathcal{B}_1, \ldots, \mathcal{B}_n))$ with the initial valuation $\mathbf{X}^{\mathbf{1}}_{\mathbf{init}} \sqcup \cdots \mathbf{X}^{\mathbf{n}}_{\mathbf{init}}$. Then there exists $\mathcal{R} \subseteq S \times Q$ a symmetric relation such that*

- *$(s_0, q_0) \in \mathcal{R}$;*
- *if $(s, q) \in \mathcal{R}$ then for all $s' \in S$ and $s \xrightarrow[l]{[n]} s' \in T$ there exists $q' \in Q$ and $q \xrightarrow{p} q' \in \pi$ with $\mathbb{P}(q \xrightarrow{p} q') = n$ such that $(s', q') \in \mathcal{R}$.*

Lack of space prevents us from writing the proof here, but it is available in the appendix. Moreover, the proof is a straightforward, albeit tedious, application of the definitions. As we noted at the beginning of the section, the correspondence between IoT processes and their $\mathcal{S}$BIP components is kept throughout the transformation thanks to the labeling functions. The stochastic behavior of $\mathcal{S}$BIP is implemented in IoT by the Choice rule. Lastly priorities in $\mathcal{S}$BIP are implemented by the rules ParState_Tau and ParState_Interaction.

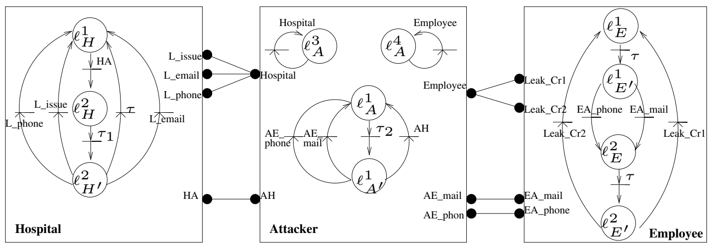


Fig. 6: Transformation of the Smart-Hospital example

As an example, we show in Figure 6 the transformation of the IoT model of the Smart Hospital in an $\mathcal{S}$BIP component.

6 Evaluating the Probability of an Attack

In this section we use executions of an IoT system to evaluate the probability of an attack. Thanks to Theorem 1, instead of reasoning on an IoT system, we can use the corresponding $\mathcal{S}$BIP system.

We employ two SMC techniques. We first use the Monte Carlo method, which consists of sampling executions and then estimating the probability of an attack, based on the number of executions for which the attack was successful. However, as we will see in the next section, the Monte Carlo method requires a large number of simulations for a correct estimate of an event which occurs with probability 10^{-5}. The experimental framework we used does not scale well for a large number of simulations. We therefore employed a second SMC technique, called *importance splitting* [9]. This technique is tailor-made for *rare* events, that is precisely events that occur rarely in a simulation, and for which Monte Carlo does not scale.

Importance splitting requires the decomposition of an execution leading to an attack into a sequence of elements, called levels and denoted l_i, for $i \leq m$ and for a decomposition in m levels. The first level is reached by all executions, while the last level is reached only if the attack succeeds. The levels are ordered $l_0 < \cdots < l_m$ meaning that level l_i is reached only if the previous levels $l_{j<i}$ have been reached before. We write $\mathbb{P}(\sigma > l_i)$ for the probability that l_i was reached during an execution σ. Then $\mathbb{P}(\sigma > l_i) = \mathbb{P}(\sigma > l_i \mid \sigma > l_{i-1})\mathbb{P}(\sigma > l_{i-1})$. Therefore we can compute the probability of the attack as follows: $\prod_{i=1}^{n} \mathbb{P}(\sigma > l_i \mid \sigma > l_{i-1})$. To infer the levels, importance splitting uses a *score* function defined on executions. Intuitively the closer we get to a successful attack, the higher the score.

Attack trees provide an initial decomposition of the attack, on which the score function is defined. The attack tree is transformed into a $\mathcal{S}$BIP component, called a *monitor*. The leaves of the tree are some of the interactions between the Attacker and the other components in the model. The branches of the tree are internal transitions to the monitor component. In a monitor obtained from an attack tree t we associate a Boolean variable, denoted v_{n}, for every node (or leaf) n of t. The variable associated to a leaf is set to *true* when the associated event occurred in the monitored execution. The variables of each other node are updated according to their corresponding Boolean expression.

We write $h(\mathsf{t})$ for the height of a tree t and $d(\mathsf{n}, \mathsf{t})$ for the depth of node n in t. The score of an execution is computed as $\mathsf{score} = h(\mathsf{t}) - d(\mathsf{n}, \mathsf{t})$, where n is the highest node for which v_{n} is *true*.

Definition 11 (Monitor). *The monitor $M_{\mathsf{t}} = (P, V, N)$ of an attack tree t is defined as follows:*

- $P = \{p_{SR}, p_{LC}, p_{score}\}$ *consists of two ports used for observing the SR and LC interactions and of a third port used for an internal transition that updates the score;*
- $V = V^d \cup V^p$ *where* $V^d = \{\mathsf{score}\} \cup \{v_{\mathsf{n}} \mid \mathsf{n}$ *is a node of* $\mathsf{t}\}$ *and* $V^p = \emptyset$*;*
- $N = (\{l_0\}, \{l_0\}, T)$ *is a Petri-Net with only one place* l_0 *and with* $T = \{(l_0, \langle p, \mathit{true}, f\rangle, l_0) \mid p \in P\}$ *where* f *updates the variables* v_{n} *and* score*.*

For each interaction in an $\mathcal{S}$BIP system, we add a port of the monitor to the interaction. In this manner, the monitor can observe the system and update its Boolean variables accordingly.

Definition 12. *Let Γ be a set of interactions of a $\mathcal{S}$BIP system.* The set of interactions Γ' between the $\mathcal{S}$BIP system and a monitor $M_{\mathsf{t}} = (P, V, N)$ *is defined as follows: for all* $\gamma = (\{a, a'\}, G_\gamma, F_\gamma) \in \Gamma$, *let* $(\{a, a', p\}, G_\gamma, F_\gamma \sqcup f) \in \Gamma'$ *where p and f are defined as follows:*

- *if* $a = e_1 \xrightarrow[v]{c} e_2$ *then* $p = p_{SR}$ *and* $f = \{v_{\mathsf{n}} := \mathit{true},$ *if* $v_{\mathsf{n}} \in V\}$, *for* $\mathsf{n} = (SR, v)$*;*
- *if* $a = e_1 \xrightarrow[v]{} \!\!\!\!\!\rightarrow e_2$ *then* $p = p_{SR}$ *and* $f = \{v_{\mathsf{n}} := \mathit{true},$ *if* $v_{\mathsf{n}} \in V\}$ *for* $\mathsf{n} = (LC, v)$*;*

7 Implementation and Experiments

In this section we describe the tool chain we implemented and some experiments based on the Smart Hospital example. We provide all resources necessary for replicating the experiments at `http://iot-modeling.gforge.inria.fr`.

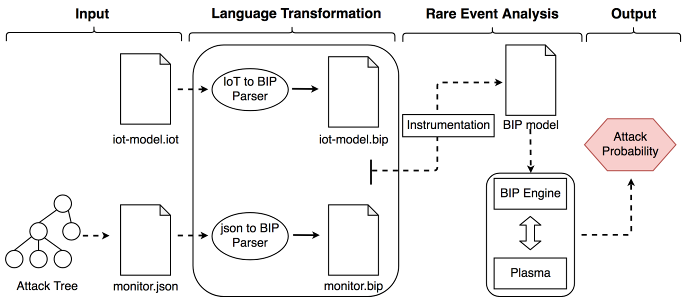


Fig. 7: Tool-set Overview

In the diagram of Figure 7, we describe the tool chain we implemented. The user provides an IoT system and an attack tree in the form of a *json* file. We implemented two parsers, "IoT-to-BIP" and "json-to-BIP", that transform the IoT model and the attack tree into two $\mathcal{S}$BIP files. The $\mathcal{S}$BIP files are compiled into a BIP executable. The BIP simulation engine runs the executable and interacts with Plasma [5], the statistical model checker we used.

The results of our experiments are shown in Figure 8. The model we used is based on our running example, the Smart Hospital.

We used two variants of our IoT model to calculate the probabilities of the success of an attack using the Monte Carlo and the importance splitting methods. The results and times presented in Figure 8 are obtained by averaging outcomes of 10 iterations. We observe that importance splitting gives a correct estimate from 1000 simulations, whereas the Monte Carlo method needs 100 times (1000 times for Model B) more simulations. However the importance splitting methods does not behave well when running on a large number of simulations. Therefore we argue that both methods are useful and complement each other in our analysis: Monte Carlo for estimating a probability n when we can produce around

	Model A				Model B			
	Monte Carlo		Importance Splitting		Monte Carlo		Importance Splitting	
Nb of Simulations	Result ($\times 10^{-5}$)	Time (s)	Result ($\times 10^{-5}$)	Time (s)	Result ($\times 10^{-5}$)	Time (s)	Result ($\times 10^{-6}$)	Time (s)
1000	0	6	6,7	8	0	6	6,4	8
10000	8,0	15	5,2	30	0	15	11,9	30
100000	6,7	255	7,0	1349	0	251	5,8	1088
1000000	6,4	10410	4,0E-5	26330	7,5	10423	4,3E-6	27013

Fig. 8: Experiments: In model B the *leaks* are twice less probable than in model A.

$10/n$ simulations, and importance splitting for experiments with fewer simulations.

8 Related Work

In this section we compare our work with some other works in the literature. First we explain why we defined our own formal language for IoT. Then we look at other works that used attack trees and model checking to analyse security properties of a system.

The formal language we propose for specifying IoT systems is an extension of [3] to probabilistic systems, inspired by the probabilistic CCS [17]. However our treatment of probabilities is slightly different (see rule Choice in Figure 2). An IoT model in our language is transformed into a $\mathcal{S}$BIP system, which has a given semantics. The operational semantics we defined therefore reflects the way probabilities are handled in $\mathcal{S}$BIP. Moreover, there are also some difference in terms of expressiveness between CCS and our language. We designed our language such that entities can store data (and thus have some notion of state) and we distinguish between two types of interactions: either safe ones or interactions that contain leaks of sensitive data. We motivate these features of our language in Section 2.

Attack trees are used extensively for modeling security attacks, including internet attacks as in [16]. In this work we use attack trees in combination with a model checker to monitor a system's execution and infer the probability of a successful attack. We therefore restrict our survey to related works that do a similar analysis.

In [11, 7] attack trees are translated into timed and stochastic automata. Model checking is then used to infer various properties of an attack, such as its probability of success or its cost. In [8] attack trees are translated into stochastic Petri Nets such that attacks can then be simulated. Compared to these works, we model both the attacks, using the attack trees, *and* the system on which the attacks are carried. In our approach it is the model of the IoT system that is decorated with probabilities and is simulated, and not the attack trees. Moreover, the tools and the approach we propose can also tackle rare events in security issues.

The authors of [13] presented a formal model to describe IoT systems and ensure its functional correctness. Their proposed approach is similar to ours, however the language we propose is more general and, similar to above, we allow for a stochastic analysis that handles rare attacks.

In [14] the *importance sampling* technique for rare events is applied to *fault-trees*, a variant of attack trees. The technique consists of computing a new distribution that makes the rare event more frequent. The results are then adjusted w.r.t. the difference between the normal distribution and the importance sampling one. Our method based on importance splitting, is better suited for attack trees, as the intermediate steps leading to a rare event correspond to the nodes in the tree leading to a successful attack. Moreover, there are no additional steps as in the case of importance sampling.

Formal semantics for attack trees have been proposed in [12, 10] and we draw inspiration from them when introducing attack trees in Section 3.

9 Conclusion

In this paper, we proposed a sound probabilistic framework for modeling IoT systems and verifying its security using attack trees. The approach consists on transforming a high level IoT model and its attack tree into a $\mathcal{S}$BIP model. We showed on a complex example how to estimate the probability of success of an attack using SMC techniques.

Acknowledgements. We would like to thank Axel Legay for his helpfull suggestions on importance splitting, and Jean Quilbeuf for his technical help in the tool implementation.

References

[1] *Anatomy of an Attack, MEDJACK (Medical Device Attack).* Tech. rep. May 2015.

[2] Ananda Basu, Marius Bozga, and Joseph Sifakis. "Modeling Heterogeneous Real-time Components in BIP". In: *SEFM.* 2006. DOI: `10.1109/SEFM.2006.27`.

[3] Delphine Beaulaton et al. "A Language for Analyzing Security of IOT Systems". In: *SoSE.* 2018. DOI: `10.1109/SYSOSE.2018.8428704`.

[4] Saddek Bensalem et al. "Statistical Model Checking Qos Properties of Systems with SBIP". In: *ISoLA.* 2012. DOI: `10.1007/978-3-642-34026-0_25`.

[5] Benoît Boyer et al. "PLASMA-lab: A Flexible, Distributable Statistical Model Checking Library". In: *QEoS.* 2013. DOI: `10.1007/978-3-642-40196-1_12`.

[6] ENISA. *Smart Hospitals, Security and Resilience for Smart Health Service and Infrastructures.* Tech. rep. 2016.

[7] Olga Gadyatskaya et al. “Modelling Attack-defense Trees Using Timed Automata”. In: *FORMATS 2016 -14th International Conference on Formal Modelling and Analysis of Timed Systems*. Quebec City, Canada, 2016, pp. 35–50. URL: https://hal.inria.fr/hal-01406706.

[8] George C. Dalton II et al. “Analyzing Attack Trees using Generalized Stochastic Petri Nets”. In: *2006 IEEE Information Assurance Workshop*. 2006. DOI: 10.1109/IAW.2006.1652085.

[9] Cyrille Jegourel, Axel Legay, and Sean Sedwards. “Importance Splitting for Statistical Model Checking Rare Properties”. In: *CAD*. 2013. DOI: 10.1007/978-3-642-39799-8_38.

[10] Barbara Kordy, Marc Pouly, and Patrick Schweitzer. “Computational Aspects of Attack–Defense Trees”. In: *Security and Intelligent Information Systems*. 2012.

[11] Rajesh Kumar et al. “Effective Analysis of Attack Trees: A Model-Driven Approach”. In: *Fundamental Approaches to Software Engineering*. 2018.

[12] Sjouke Mauw and Martijn Oostdijk. “Foundations of Attack Trees”. In: *Proceedings of the 8th International Conference on Information Security and Cryptology*. ICISC’05. Seoul, Korea: Springer-Verlag, 2006, pp. 186–198. ISBN: 3-540-33354-1, 978-3-540-33354-8. DOI: 10.1007/11734727_17. URL: http://dx.doi.org/10.1007/11734727_17.

[13] Samir Ouchani. “Ensuring the Functional Correctness of IoT through Formal Modeling and Verification”. In: *Model and Data Engineering*. Ed. by El Hassan Abdelwahed et al. Cham: Springer International Publishing, 2018.

[14] Enno Ruijters et al. “Rare Event Simulation for Dynamic Fault Trees”. In: *CSRS*. 2017. DOI: 10.1007/978-3-319-66266-4_2.

[15] Bruce Schneier. *Secrets & Lies: Digital Security in a Networked World*. 2000.

[16] Terry Tidwell et al. “Modeling Internet Attacks”. In: *IA*. 2001.

[17] R.J. Vanglabbeek, S.A. Smolka, and B. Steffen. “Reactive, Generative, and Stratified Models of Probabilistic Processes”. In: *Information and Computation* 121 (1995). DOI: https://doi.org/10.1006/inco.1995.1123.

Appendix

Counting functions for the Operational Semantics of IoT

Definition 13 (Counting τ transitions from a state). *The functions* $\mathsf{count}_\tau : \mathit{State} \to \mathbb{N}$ *and* $\mathsf{count\_proc}_\tau : \mathit{Proc} \to \mathbb{N}$ *are defined as follows:*

$$\begin{aligned}
\mathsf{count}_\tau(s|t) =& \mathsf{count}_\tau(s) + \mathsf{count}_\tau(t)\\
\mathsf{count}_\tau(\langle P, k\rangle) =& \mathsf{count\_proc}_\tau(P)\\
\mathsf{count\_proc}_\tau(0) =& 0\\
\mathsf{count\_proc}_\tau(\alpha.P) =& 1 \ \textit{if } \alpha \neq \tau\\
& 0 \ \textit{if } \alpha = \tau\\
\mathsf{count\_proc}_\tau(\sum \alpha_i.P_i) =& 1\\
\mathsf{count\_proc}_\tau(P \mid Q) =& \mathsf{count\_proc}_\tau(P) + \mathsf{count\_proc}_\tau(Q).
\end{aligned}$$

For counting the number of interactions, we have first to rewrite a state into a *canonical* form:

$$s \equiv s_S \mid s_R \mid s_L \mid s_C \qquad \text{where} \qquad \begin{aligned}
s_S =& \langle P_1^{\mathrm{S}}, k_1^{\mathrm{S}}\rangle \mid \cdots \langle P_{nS}^{\mathrm{S}}, k_{nS}^{\mathrm{S}}\rangle\\
s_R =& \langle P_1^{\mathrm{R}}, k_1^{\mathrm{R}}\rangle \mid \cdots \langle P_{nR}^{\mathrm{R}}, k_{nR}^{\mathrm{R}}\rangle\\
s_L =& \langle P_1^{\mathrm{L}}, k_1^{\mathrm{L}}\rangle \mid \cdots \langle P_{nL}^{\mathrm{L}}, k_{nL}^{\mathrm{L}}\rangle\\
s_C =& \langle P_1^{\mathrm{C}}, k_1^{\mathrm{C}}\rangle \mid \cdots \langle P_{nC}^{\mathrm{C}}, k_{nC}^{\mathrm{R}}\rangle
\end{aligned}$$

and where $P_i^{\mathrm{S}} \equiv a.P$ and the action a is a send; nS is the number of processes of the form above in s. Similarly we define the rest of the processes. Note that if we cannot rewrite a state in this form then the rule ParState_Interaction cannot be applied (any internal or sum transitions have priority over the interactions). Moreover entities can only communicate with other entities, that is interactions are not defined internally to an entity. We therefore only need to count interactions between entities.

The function $\mathsf{count}_{\mathrm{SR,LC}}$ uses an auxiliary function $\bar{\cdot}$: action $\to$ action which defines an action $\overline{a}$ which can synchronise with a using the rules SendReceive or LeakCollect.

Definition 14. *Let* $s \equiv s_S \mid s_R \mid s_L \mid s_C$ *be a state in a canonical form. The function* $\mathsf{count}_{SR,LC} : \mathit{State} \to \mathbb{N}$ *is defined on* s *as follows:*

$$\begin{aligned}
\mathsf{count}_{SR,LC}(s_S \mid s_R \mid s_L \mid s_C) &= \mathsf{count}_{SR}(s_S, s_R) + \mathsf{count}_{LC}(s_L \mid s_C)\\
\mathsf{count}_{SR}(\langle a.P, k\rangle \mid s, t) &= \mathsf{count}(a, t) + \mathsf{count}_{SR}(s, t)\\
\mathsf{count}_{LC}(\langle a.P, k\rangle \mid s, t) &= \mathsf{count}(a, t) + \mathsf{count}_{LC}(s, t)\\
\mathsf{count}(a, \langle b.P, k\rangle \mid t) &= 1 + \mathsf{count}(a, t) \ \textit{if } a = \overline{b}\\
&= \mathsf{count}(a, t) \ \textit{otherwise}
\end{aligned}$$

Proof of Theorem 1

Lemma 1. *Any two congruent IoT states have the same transformation in $\mathcal{S}BIP$ systems.*

Proof. We proceed by cases on the congruence relation. First consider the congruence relation on states: For the monoid laws on $|$, note that the transformation results in a set of atomic components and therefore the order of states in the parallel composition does not matter. In the case where processes are congruent, we distinguish two subcases: (i) Threads in a parallel composition translate into tuples of states in the transformation of a process (Definition 9) where the order of the states does not matter; (ii) For the rest we use the fact that inside an atomic component the states that have congruent labels are identified.

Proof (Theorem 1). Let $e_1, \cdots, e_n$ be n entities of an IoT system (S, L, T, s_0) with the initial states $\langle P_1, k_1\rangle, \cdots \langle P_n, k_n\rangle$. $\mathcal{B}_{e_i} = (P_i, V_i, N_i)$ with $N_i = (L_i, L_{i,0}, T_i, F_i)$, is the transformation of the current state of the entity e_i, for $i \leq n$. Also let $V_i = V_i^p \cup V_i^d$. We write (P, Q, π, q_0) for the semantics of $\langle\ll\rangle\Gamma(\mathcal{B}_{e_1}, \cdots \mathcal{B}_{e_n}) = (\Gamma, \mathbf{V}, \mathbf{N})$ with $\mathbf{N} = (\mathbf{L}, \mathbf{L_0}, \mathbf{T}, \mathbf{F})$. Lastly $\mathbf{X_{init}}$ is the initial valuation.

To construct the relation $\mathcal{R} \subseteq S \times Q$ required by the theorem, we first set some notations and constraints below. Informally, these constraints establish the relation between the processes and knowledge functions in states of S and the markings and the valuations, respectively, in states of Q.

1. **Correspondence between Processes and Markings.** For a thread T let us write m_T for the marking associated with T and defined as follows:
$$m_T(l) = \begin{array}{ll} 1 \text{ if } \ell(l) = T \text{ or } \ell(l) = U^\star, U = [n]T + T', & \text{for some threads } T', U \\ 0 \text{ otherwise} & \end{array}$$
where (P, V, N) and $N = (\mathcal{L}, \mathcal{L}_0, \mathcal{T}, \mathcal{F})$ is obtained as in Definition 8 and where $l \in \mathcal{L}$. For a process $P = T_1 \mid \cdots \mid T_m$, let us write m_P for the marking associated with P and defined as $m_{T_1} + \cdots + m_{T_m}$.
2. **Correspondence between Knowledge and the Deterministic Variables.** From Definition 8 it follows that for each thread T_j in a process P_i we define the set $V_i^d = \{v_c \mid c \text{ is a protocol used in } T_j\}$. From Definition 9 then the set of variables of $P_i = T_1 \mid \cdots \mid T_m$ is $\cup_{j \leq m} V_j = \{v_c \mid c \text{ is a protocol used in } P_i\}$.
Then, if $\mathbf{X_i}$ the current valuation of entity e_i, we require that $\mathbf{X_i}(v_c) = k_i(c)$, for $i \leq n$, $c \in C$ and $v_c \in V_i^d$. Recall that we write C for the set of protocols used in the IoT system and k_i for the knowledge function of an entity e_i.
3. **Correspondence between Probabilistic Choices in Processes and the Random Variables.** For every summation thread U in a process P_i, we have that there exists a random variable $v_U \in V_i^p$, by Definition 8. Moreover, if T a thread of P_i, belongs to a summation, i.e. $U = [n]T + T'$, for some threads T', U, then for the current valuation $\mathbf{X_i}$ we have that $\mathbf{X_i}(v_U) = T$. For a process $P = T_1 \mid \cdots \mid T_m$ we use Definition 9 and have that V^p is the disjoint union of all V_j^p, where V_j^p is the set of random variables for T_j, $j \leq m$.

We define the following relation between the states of S and the states of Q:

$$\mathcal{R} = \big\{ (\langle P_1, k_1 \rangle \mid \cdots \mid \langle P_n, k_n \rangle, (m = m_{P_1} + \cdots m_{P_n}, \mathbf{X} = \mathbf{X_1} \sqcup \cdots \sqcup \mathbf{X_n})) \mid \\ \text{the conditions 1-3 above hold} \big\}.$$

We show that $\mathcal{R}$ is the relation required in Theorem 1. First we have to show that $(s_0, q_0) \in \mathcal{R}$.

We use Definition 8 from which we have that $\mathcal{L}_0 = \{l_T\}$ is the initial place in the transformation of a thread T. Then, by Definition 9, $L_0 = \uplus_{j \leq m} \mathcal{L}_0^j = \uplus_{j \leq m} \{l_{T_j}\}$ is the initial set of places in the transformation of a process $P = T_1 \mid \cdots \mid T_m$. From Definition 7 it follows that $\mathbf{L}_0 = \uplus_{i \leq n} L_{0,i}$. By Definition 5 the initial marking in $q_0 = (m_0, \mathbf{X_{init}})$ is defined as $m_0(l) = 1 \iff l \in \mathbf{L}_0$ and 0 otherwise. Hence we can write $m_0 = m_{P_1} + \cdots + m_{P_n}$. This shows condition 1 of $\mathcal{R}$.

From Definition 9 we have that for each entity e_i, $\mathbf{X_{init}}(v_c) = k_i(c)$, for all protocols c used by e_i. From Definition 7 the set of variables of the composed $\mathcal{B}_{e_i}$ components is the disjoint union V_i, i.e. $\mathbf{V} = \uplus_{i \leq n} V_i$, in particular $\mathbf{V}^d = \uplus_{i \leq n} V_i^d$. Then a valuation for $\mathbf{V}$ is the disjoint composition of the individual valuations for V_i, from which it follows the required decomposition of $\mathbf{X_{init}}$ in $q_0 = (m_0, \mathbf{X_{init}})$. Therefore condition 2 of $\mathcal{R}$ holds. For condition 3 to hold suffices to note that there is no probabilistic choice made yet in any process and therefore there is no correspondence to show. We can take any initial valuation we want for the random variables.

Let us now suppose that $(s, q) \in \mathcal{R}$ and that $s \xrightarrow[l]{[n]} s'$, for some label $l \in L$, some probability n and state $q' \in Q$. We have to show that there exists $q' \in Q$ and $q \xrightarrow{p} q' \in \pi$ with $\mathbb{P}(q \xrightarrow{p} q') = n$ such that $(s', q') \in \mathcal{R}$. We reason by cases on the label l of the transition $s \xrightarrow[l]{[n]} s'$.

- Let $l = SR : v$ or $l = LC : v$; then let e_1 and e_2 be the two communicating entities. Using Lemma 1 we can rewrite the transition as follows:

$$s = \langle P_1, k_1 \rangle \mid \langle P_2, k_2 \rangle \mid \langle P_3, k_3 \rangle \mid \ldots \mid \langle P_n, k_n \rangle \xrightarrow[l]{[1/m]} \\ s' = \langle Q_1, k_1' \rangle \mid \langle Q_2, k_2' \rangle \mid \langle P_3, k_3 \rangle \mid \ldots \mid \langle P_n, k_n \rangle$$

  where we can decompose $P_1 \equiv_P a_1.T_1 \mid P_1'$ and $P_2 \equiv_P a_2.T_2 \mid P_2'$, $Q_1 \equiv_P T_1 \mid P_1'$ and $Q_2 \equiv_P T_2 \mid P_2'$, again by Lemma 1 and from the rules of Figure 2. Here we suppose w.l.o.g. that a_1 and a_2 are the two synchronizing actions in P_1 and P_2, respectively. Also suppose w.l.o.g. that a_1 is a send (or a leak) and that a_2 is a receive (or a collect). Let c be the protocol used for the communication in case $l = SR : v$.
  From $(s, q) \in \mathcal{R}$ we have that $q = (m_{P_1} + \cdots m_{P_n}, \mathbf{X_1} \sqcup \cdots \sqcup \mathbf{X_n})$ and that $m_{P_i} = m_{a_i.T_i} + m_{P_i'}$, for $i \leq 2$. Also from condition 1 of $\mathcal{R}$, $m_{a_i.T_i} = \{l_i\}$ with either $\ell(l_i) = a_i.T_i$, or $\ell(l_i) = U_i^\star$, for some summation threads U_1, U_2.

- If $\ell(l_1) = a_1.T_1$ then we use the transformation of Definition 8 to show that there exists the place $l'_1 \in L_1$, with $\ell(l'_1) = T_1$ and the transition $t_1 = (\{l_{a_1.T_1}\}, \langle a_1, g_1 = \mathit{true}, f_1\rangle, \{l_{T_1}\})$ in $\mathcal{B}_1$.
  - ∗ If $\ell(l_2) = a_2.T_2$ then as above, there exists $l'_2 \in L_2$, with $\ell(l'_2) = T_2$ and the transition $t_2 = (\{l_{a_2.T_2}\}, \langle a_2, g_2 = \mathit{true}, f_2\rangle, \{l_{T_1}\})$ in $\mathcal{B}_2$.
  - ∗ $\ell(l_2) = U_2^\star$, with $U_2 = [n_2]a_2.T_2 + U'_2$, for some threads U_2, U'_2. As in the case above, from Definition 8 we have that there exists the places $l'_2 \in L_2$ with $\ell(l'_2) = T_2$. We also have, from condition 3 of $\mathcal{R}$ that there exists a random variable $v_{U_2} \in V_2^p$ with $\mathbf{X}(v_{U_2}) = a_2.T_2$. Moreover we have the transition $t_2 = (\{l_{U_2^\star}\}, \langle a_2, g_2 = (v_{U_2} == a_2.T_2), f_2\rangle, \{l_{T_2}\})$ in $\mathcal{B}_2$.
- the other case is similar.

Note that in all cases above, $f_i = \{v := v \mid v \in V^d\}$ with $R_i^p = \emptyset$, $i \leq n$.
Using Definition 10 we have that there exists an interaction $\gamma = (\{a_1, a_2\}, G, F)$ such that
- If $l = SR : v$ then $G = (\exists x \in v_c^1$ such that $x \in v_c^2)$ for $v_c^1 \in V_1$ and $v_c^2 \in V_2$.
- If $l = LC : v$ then $G = \mathit{true}$.

Also, $F = \{v_{c'}^2 := v_{c'}^2 \cup \{v'\} \mid \mathsf{protocol}(v') = c', v_{c'}^2 \in V_2\}$ for both $l = SR : v$ and $l = LC : v$.
We now use Definition 7 and have that there exists the transition

$$\underline{T} = (\{l_1, l_2\}, \langle \gamma, g_1 \wedge g_2 \wedge G, (f_1 \sqcup f_2) \circ F\rangle, \{l'_1, l'_2\}) \in \mathbf{T}.$$

We have to show that the guard $g = g_1 \wedge g_2 \wedge G$ holds for the current valuation $\mathbf{X}$:
- If $g_1 = (v_{U_1} == a_1.T_1)$ then $\mathbf{X}(g_1)$ holds from condition 3 of $\mathcal{R}$; otherwise $g_1 = \mathit{true}$. We proceed similarly for g_2.
- If $l = SR : v$ then $G = (\exists x \in v_c^1$ such that $x \in v_c^2)$ for $v_c^1 \in V_1$ and $v_c^2 \in V_2$. From condition 2 of $\mathcal{R}$ we have that $\mathbf{X}(v_c^i) = k_i(c)$, $i \leq n$. Then the guard holds as it is the condition of rule SendReceive in Figure 2. If $l = LC : v$ then $G = \mathit{true}$.

The transitions above are allowed to proceed by the priority order $\ll$ (see Definition **??**) only if there is no internal transition available. This is the case as ensured by the rule ParState_Interaction in Figure 2.
Therefore, by Definition 5, there exists the transition

$$q = (m_{P_1} + m_{P_2} + \cdots m_{P_n}, \mathbf{X_1} \sqcup \cdots \sqcup \mathbf{X_n}) \xrightarrow{\gamma} q' = (m', \mathbf{X}')$$

where we have to show that conditions 1-3 of $\mathcal{R}$ hold. For condition 1 we have to show that $m' = m_{Q_1} + m_{Q_2} + \cdots m_{P_n}$. Using Definition 5 it follows that

$$m' = m - ^\bullet \underline{T} + \underline{T}^\bullet = m - \{l_1, l_2\} + \{l'_1, l'_2\}.$$

As $\mathbf{L}_0 = \uplus_{i \leq n} L_{0,i}$, from Definition 7, it follows that

$$m' = (m_{P_1} - \{l_1\} + \{l'_1\}) + (m_{P_2} - \{l_2\} + \{l'_2\}) + \cdots + m_{P_n}.$$

Using condition 1 of $\mathcal{R}$ on m_{P_1} and m_{P_2} we have that $m_{P_1} - \{l_1\} + \{l_1'\} = m_{Q_1}$ and similarly for m_{Q_2}.
Let us now show condition 2, i.e. $\mathbf{X'} = \mathbf{X_1'} \sqcup \mathbf{X_2'} \sqcup \cdots \sqcup \mathbf{X_n}$ and $\mathbf{X_i'}(v_{c'}) = k_i'(c')$, $i \leq 2$. Using the function F above we have that $\mathbf{X_i'}(v_{c'}) = \mathbf{X_i}(v_{c'}) \cup \{v\}$. From rules SendReceive and LeakCollect we also get that $k_i'(c') = k_i(c) \cup \{v\}$, $i \leq 2$.
As $R_1^p = R_2^p = \emptyset$ condition 3 is trivial.
Lastly, the two transitions have the same probability: $|\mathsf{Enabled}(m; \mathbf{X})| = m$ by Lemma **??**, and therefore $\mathbb{P}\big(q \xrightarrow{p} q'\big) = 1/m$.

– Let $l = \tau$; let e_1 be the entity that triggers the internal transition. Using Lemma 1 we can rewrite the states in the transition as follows:

$$s = \langle P_1, k_1 \rangle \mid \langle P_2, k_2 \rangle \mid \langle P_3, k_3 \rangle \mid \ \ldots \ \mid \langle P_n, k_n \rangle \xrightarrow[l]{[n]}$$
$$s' = \langle Q_1, k_1' \rangle \mid \langle P_2, k_2 \rangle \mid \langle P_3, k_3 \rangle \mid \ \ldots \ \mid \langle P_n, k_n \rangle.$$

There are two possibilities: either $P_1 \equiv_P \sum_{i \in I_1} a_i.T_i \mid P_1'$ where $Q_1 = a_1.T_1$ w.l.o.g. or $P_1 \equiv_P \tau.T_1 \mid P_1'$ with $Q_1 = T_1$. We write $U = \sum_{i \in I_1} a_i.T_i$ or $U = \tau.T_1$ depending on which of the two cases we are.
From $(s, q) \in \mathcal{R}$ we have that $q = (m_{P_1} + \cdots m_{P_n}, \mathbf{X_1} \sqcup \cdots \sqcup \mathbf{X_n})$ and that $m_{P_1} = \{l\} + m_{P_1'}$, $\ell(l) = U$. We use the transformation of Definition 8 to show that there exists the place $l' \in L_1$ and the transition $t = (\{l\}, \langle \tau, g = \mathit{true}, f \rangle, \{l'\})$ in $\mathcal{B}_1$.

- If $U = \sum_{i \in I_1} a_i.T_i$ then $\ell(l') = U^\star$, $f = \{v := v \mid v \in V_1^d\}$ and $R^p = \{v_U\}$.
- If $U = \tau.T_1$ then $\ell(l') = T_1$, $f = \{v := v \mid v \in V_1^d\}$ and $R^p = \emptyset$.

Using Definition 10 we have that there exists an interaction $\gamma = (\{\tau\}, G = \mathit{true}, F)$ with $F = \{v := v \mid v \in V_1^d\}$.
From Definition 7 there exists the transition

$$\underline{T} = (\{l\}, \langle \gamma, g_1 \wedge G = \mathit{true}, f \circ F \rangle, \{l'\}) \in \mathbf{T}.$$

The guard trivially holds and we obtain the transition

$$q = (m_{P_1} + + \cdots m_{P_n}, \mathbf{X_1} \sqcup \cdots \sqcup \mathbf{X_n}) \xrightarrow{\gamma} q' = (m', \mathbf{X'})$$

where we have to show that conditions 1-3 of $\mathcal{R}$ hold. As in the first case, condition 1 follows from $m' = m - \{l\} + \{l'\} = m_{Q_1} + \cdots m_{P_n}$. Condition 2 trivially hold as the update functions f and F are the identity and therefore $\mathbf{X_1}' = \mathbf{X_1}$. Indeed the knowledge function of k_1 is not modified by the rules Choice or Internal.
To show condition 3 we use Definition 8 from which we have that there exists $v_U \in V_1^p$, $v_U \sim \mu$, where $\mu(a_1.T_1) = n_1$. Then we can take $\mathbf{X'}(v_U) = a_1.T_1$. We also this argument to show that the two transitions have the same probabilities: by Lemma **??**, $|\mathsf{Enabled}(m; \mathbf{X})| = m$ and therefore $\mathbb{P}\big(q \xrightarrow{p} q'\big) = 1/m \times n_1$.

Hereafter we prove the similarity of the IoT system to its corresponding $\mathcal{S}$BIP model. Let us suppose that $(q, s) \in \mathcal{R}$ and that $q \xrightarrow{\gamma} q'$, for a transition labelled with γ, where $q, q' \in Q$. We have to show that there is a state $s' \in S$ with $s \xrightarrow[l]{[n]} s'$, for some label $l \in L$, such that $(s', q') \in \mathcal{R}$. We define $s = \langle P_1, k_1\rangle \mid \langle P_2, k_2\rangle \mid \cdots \mid \langle P_n, k_n\rangle$. Here we also reason by cases: whether the transition is an interaction between two components $\mathcal{B}_{e_1}$ and $\mathcal{B}_{e_2}$ or an internal transition.

– We consider the communication is an interaction $\gamma = (\{a_1, a_2\}, G, F)$ between $\mathcal{B}_{e_1}$ and $\mathcal{B}_{e_2}$:

$$q = (m_{P_1} + m_{P_2} + \cdots m_{P_n}, \mathbf{X_1} \sqcup \mathbf{X_2} \sqcup \cdots \sqcup \mathbf{X_n}) \xrightarrow{\gamma} q' = (m', \mathbf{X'})$$

As it is an interaction between two entities, from Definition 7 we have that there exists the transitions $t_i = (m_i, \langle p_i, g_i, f_i\rangle, m'_i) \in T_i$, for $i \in \{1, 2\}$. From the Definition 10, $m_i = m_{P_i}$, $p_i = a_i$, $g_i = \mathit{true}$ and f_i are the constant update functions. From $(q, s) \in \mathcal{R}$ we have that $m_{P_1} = m_{a_1.T_1} + m_{P'_1}$, $m_{P_2} = m_{a_2.T_2} + m_{P'_2}$ with $P_1 = a_1.T_1 \mid P'_1$ and $P_2 = a_2.T_2 \mid P'_2$. Moreover, from the Definition 5 there exists the transition

$$\underline{T} = (m_{P_1} + m_{P_2}, \langle\{a_1, a_2\}, g_1 \wedge g_2 \wedge G, (f_1 \sqcup f_2) \circ F\rangle, m_{Q_1} + m_{Q_2}) \in \mathbf{T}$$

with $m_{Q_1} = m_{T_1} + m_{P'_1}$, $m_{Q_2} = m_{T_2} + m_{P'_2}$.
We distinguish between the two types of interactions:
- $a_1 = e_1 \xrightarrow[\text{v'}]{\text{c}} e_2$ and there exists $a_2 \in \mathsf{Actions}$ such that $a_2 = e_2 \xleftarrow{\text{c}} e_1$,
- or $a_1 = e_1 \xrightarrow[\text{v'}]{} \!\!\!\!\!\rightarrow e_2$ and there exists $a_2 \in \mathsf{Actions}$ such that $a_2 = e_2 \twoheadleftarrow e_1$

Following the Definition 10 we have the following guards:
- if $G = (\exists x \in v_c^1$ such that $x \in v_c^2)$ for $v_c^1 \in V_1$ and $v_c^2 \in V_2$ then $l = SR$
- if $G = \mathit{true}$ then $l = LC$

We can then apply the rules SendReceive or LeakCollect from Figure 2. Hence we derive an interaction between e_1 and e_2 exists for which we have to show that conditions 1-3 of $\mathcal{R}$ hols.

$$s = \langle P_1, k_1\rangle \mid \langle P_2, k_2\rangle \mid \ \ldots \ \mid \langle P_n, k_n\rangle \xrightarrow[l]{[n]}$$
$$s' = \langle Q_1, k'_1\rangle \mid \langle Q_2, k'_2\rangle \mid \ \ldots \ \mid \langle P_n, k_n\rangle.$$

From above, it follows that $m' = m_{Q_1} + m_{Q_2} + \cdots m_{P_n}$, which is the first condition of $\mathcal{R}$.
In the interaction γ, we apply the update function $F = \{v_{c'}^2 := v_{c'}^2 \cup \{v'\} \mid \mathsf{protocol}(v') = c', v_{c'}^2 \in V_2\}$ for both $l = SR : v$ and $l = LC : v$, then $\mathbf{X'_i}(v_{c'}) = \mathbf{X_i}(v_c) \cup \{v\}$. Therefore we can write $\mathbf{X'} = \mathbf{X'_1} \sqcup \mathbf{X'_2} \cdots \mathbf{X_n}$. With the interaction $s \xrightarrow[l]{[n]} s'$, we apply rules SendReceive or LeakCollect from Figure 2 where $k'_i(c') = k_i(c) \cup \{v\}$. Hence the condition 2 hols, i.e. $\mathbf{X'_i}(v_{c'}) = k'_i(c')$. With the execution of the γ interaction, the probabilistic

distribution $R_1^p = R_2^p = \emptyset$, and from the SENDRECEIVE or LEAKCOLLECT from Figure 2 is the same, then the condition 3 trivially hols. The two transitions have the same probability: $\mathbb{P}\big(q \xrightarrow{p} q'\big) = 1/m$ by Lemma **??**, and therefore $|\mathsf{Enabled}(m;\mathbf{X})| = m$.

– We consider the transition to be an internal transition τ in component $\mathcal{B}_{e_1}$. From lemma 1 we can write the transition:

$$q = (m_{P_1} + m_{P_2} + \cdots m_{P_n}, \mathbf{X_1} \sqcup \mathbf{X_2} \sqcup \cdots \sqcup \mathbf{X_n}) \xrightarrow{\gamma} q' = (m', \mathbf{X}')$$

where $s = \langle P_1, k_1\rangle \mid \langle P_2, k_2\rangle \mid \cdots \mid \langle P_n, k_n\rangle$, from $(q,s) \in \mathcal{R}$, we distinguish two cases of transition execution:

- A probabilistic choice: $m_{P_1} = \{l\} + m_{P_1'}$ where $\ell(l) = \sum_{i\in I} [n_i] a_i.T_i$ and $P_1 = \sum_{i\in I} a_i.T_i | P_1'$. From the transformation of Definition 8, the transition

$$t = (\{l_T\}, \langle \tau, \text{true}, f^\star\rangle, \{l_{T^\star}\}) \in T_1$$

can be executed where $f^\star = (\{v := v \mid v \in V^d\}$ and $R^p = \{v_T\})$. From relations of Figure 2, there exists a CHOICE transition in IoT system such that

$$s = \langle P_1, k_1\rangle \mid \langle P_2, k_2\rangle \mid \ldots \mid \langle P_n, k_n\rangle \xrightarrow[l]{[n_1]}$$
$$s' = \langle Q_1, k_1'\rangle \mid \langle P_2, k_2\rangle \mid \ldots \mid \langle P_n, k_n\rangle$$

where $Q_1 = a_1.T_1$. Now we can verify if the conditions 1-3 of R holds. We have that $m_{Q_1} = \{\ell\}^\star$ and $m' = m_{Q_1} + m_{P_2} + \cdots m_{P_n}$. As the update function f is the identity function the condition 2 trivially hold and the knowledge $k_1' = k_1$. To show condition 3 , we note that there exists $v_{T_1} \in V_1^p$, $v_{T_1} \sim \mu$ such that $\mathbf{X}'(v_{T_1}) = a_1.T_1$. We use Definition 8 from which we have that where $\mu(a_1.T_1) = n_1$.

- An internal transition: $m_{P_1} = m_{\tau.T_1} + m_{P_1'}$ and $P_1 = \tau.T_1|P_1'$. From the transformation of definition 8, the transition $\underline{T} = (\{l_{a.T}\}, \langle a, \text{true}, f\rangle, \{l_T\})$ can be executed where $f = (\{v := v \mid v \in V^d\}$ and $R^p = \emptyset)$. From relations of Figure 2, there exists an INTERNAL transition in IoT system such that

$$s = \langle P_1, k_1\rangle \mid \langle P_2, k_2\rangle \mid \ldots \mid \langle P_n, k_n\rangle \xrightarrow[l]{[n]}$$
$$s' = \langle Q_1, k_1'\rangle \mid \langle P_2, k_2\rangle \mid \ldots \mid \langle P_n, k_n\rangle$$

where $Q_1 = T_1|P_1'$. Now we can verify if the conditions 1-3 of R holds. We have that $m_{Q_1} = m_{T_1} + m_{P_1'}$ and $m' = m_{Q_1} + m_{P_2} + \cdots m_{P_n}$.
As the update function f is the identity function the condition 2 trivially hold and the knowledge $k_1' = k_1$. Then $\mathbf{X}' = \mathbf{X_1'} \sqcup \mathbf{X_2} \sqcup \cdots \sqcup \mathbf{X_n}$. Likewise, since $R^p = \emptyset$ the condition 3 trivially holds.