The cost of securing IoT communications*

Chiara Bodei and Letterio Galletta

Dipartimento di Informatica, Università di Pisa
{chiara,galletta}@di.unipi.it

Abstract. More smart objects and more applications on the Internet of Things (IoT) mean more security challenges. In IoT security is crucial but difficult to obtain. On the one hand the usual trade-off between highly secure and usable systems is more impelling than ever; on the other hand security is considered a feature that has a cost often unaffordable. To relieve this kind of problems, IoT designers not only need tools to assess possible risks and to study countermeasures, but also methodologies to estimate their costs. Here, we present a preliminary methodology, based on the process calculus IoT-LySA, to infer quantitative measures on systems evolution. The derived quantitative evaluation is exploited to establish the cost of the possible security countermeasures.

1 Introduction

Within few years the objects we use every day will have computational capabilities and will be always connected to the Internet. In this scenario, called the Internet of Things (IoT), these "smart" devices are equipped with sensors to automatically collect different pieces of information, store them on the cloud or use them to affect the surrounding environment through actuators. For instance, our smart alarm clock can drive our heating system to prepare us a warm bathroom in the morning, while an alarm sensor in our place can directly trigger an emergency call to the closest police station. Also, in a storehouse stocking perishable food, equipped with sensors to determine the internal temperature and other relevant attributes, the refrigeration system can automatically adapt the temperature according to the information collected by sensors.

The IoT paradigm introduces new pressing security challenges. On the one hand, the usual trade-off between highly secure and usable systems is more critical than ever. On the other hand, security is considered a costly feature for devices with limited computational capabilities and with limited battery power.

Back to the refrigerator system above, an attacker can easily intercept sensors communications, manipulate and falsify data. We can resort to cryptography and

* Work supported by project PRA_2016_64 "Through the fog" (University of Pisa).

V. Biló, A. Caruso (Eds.): ICTCS 2016, Proceedings of the 17th Italian Conference on Theoretical Computer Science, 73100 Lecce, Italy, September 7–9 2016, pp. 163–176 published in CEUR Workshop Proceedins Vol-1720 at `http://ceur-ws.org/Vol-1720`

to consistency checks to prevent falsification and to detect anomalies. But is it affordable to secure all the communications? Can we still obtain a good level of security by protecting only part of communications?

IoT designers have to be selective, e.g. in choosing which packets to encrypt. To this aim, designers not only need tools to assess possible risks and to study countermeasures, but also methodologies to estimate their costs. The cost of security can be computed in terms of time overhead, energy consumption, bandwidth, and so on. All these factors must be carefully evaluated for achieving an acceptable balance among security, cost and usability of the system.

First, we introduce functions over the enhanced labels to associate costs to transitions. Here, the cost of a system is specified in term of the *time* spent for transitions, and it depends on the performed action as well as on the involved nodes. However, we can easily treat other quantitative properties, e.g. energy consumption. Intuitively, cost functions define exponential distributions, from which we compute the rates at which a system evolves and the corresponding CTMC. Then, to evaluate the performance we calculate the stationary distribution of the CTMC and the transition rewards.

Usually, formal methods provide designers with tools to support the development of systems and to reason about their properties, both qualitative and quantitative. Here, we present some preliminary steps towards the development of a formal methodology to support the analysis of the security cost in IoT systems. We aim at providing a general framework with a mechanisable procedure (with a small amount of manual tuning), where quantitative aspects are symbolically represented by parameters. Their instantiation is delayed until hardware architectures and cryptographic algorithms are fixed. By only changing these parameters designers could compare different implementations of an IoT system and could choose the better trade-off between security and costs.

Technically, we define an enhanced semantics for IoT-LySA, a process calculus recently proposed to model and reason about IoT systems [4]. IoT-LySA is equipped with a static analysis that safely approximates how data from sensors spread across the system and how objects interact each other's. Our enhanced semantics follows the methodology of [7], where each transition is associated to a cost in the style of [17,1]. Here, the cost is specified in term of the *time* spent for the transition i.e. it is the rate of the transition. From rates we mechanically derive a continuous-time Markov chains that can be analysed using standard techniques and tools [19,22]. For simplicity, here we consider a subset of IoT-LySA without actuators where intra-node communication is carried out through message passing instead of a shared store. Note that our approach can be extended to deal with other optimisation criteria, e.g. energy consumption, particularly critical in IoT systems.

Structure of the paper. In the next section, we briefly introduce the process calculus IoT-LySA. In Section 3, we present a simple function that assigns rates to transitions, and we show how to obtain the CTMC associated with a given system of nodes and how to extract performance measures from it. Concluding remarks and related work are in Section 4.

2 IoT-LySA and its Enhanced Semantics

In [4] an IoT system consists of a set of nodes that communicate through message-passing. Each node is uniquely identified by a label and is made of logical (processes) and physical (sensors and actuators) components that interact through a shared store. For simplicity, in the following we do not consider actuators, and we replace conditional construct with non deterministic choice. Furthermore, following the approach in [3] we assume a finite set of keys that are known a priori by the nodes.

Syntax. In IoT-LySA systems $N \in \mathcal{N}$ consist of a fixed number of nodes that host control processes $P \in \mathcal{P}$ and sensors $S \in \mathcal{S}$. The syntax is in Tab. 1, where $\mathcal{V}$ denotes the set of values, while $\mathcal{X}$ and $\mathcal{Z}$ are the local and the global variables, respectively, and $\mathcal{K} \subseteq \mathcal{V}$ denotes the set of cryptographic keys (e.g. K_0).

$\mathcal{N} \ni N ::=$ *systems of nodes*		$\mathcal{B} \ni B ::=$ *node components*	
$\mathsf{0}$	inactive node	P	process
$\ell : [B]$	single node	S	sensor
$N_1 \mid N_2$	composition	$B \parallel B$	composition

$P ::=$ *control processes*	
$\mathsf{0}$	nil
$\langle E_1, \cdots, E_k \rangle . P$	intra-node output
$\langle\!\langle E_1, \cdots, E_k \rangle\!\rangle \triangleright L . P$	multi-output L$\subseteq \mathcal{L}$
$(E_1, \cdots, E_j;\ x_{j+1}, \cdots, x_k) . P$	input (with match.)
$P_1 + P_2$	summation
$A(y_1, \ldots, y_n)$	recursion
$\mathsf{decrypt}\ E\ \mathsf{as}$ $\{E_1, \cdots, E_j;\ x_{j+1}, \cdots, x_k\}_{K_0}\ \mathsf{in}\ P$	decryption (with match.)
$(i; z_i).P$	clear input from sensor i
$(\{i; z_i\}_K).P$	crypto input from sensor i

$S ::=$ *sensor processes*		$E ::=$	*terms*
$\tau.S$	internal action	v	value
$\langle i, v \rangle . S$	i^{th} output	x	variable
$\langle \{i, v\}_K \rangle . S$	i^{th} enc. output	z	sensor's variable
$A(y_1, \ldots, y_n)$	recursion	$\{E_1, \cdots, E_k\}_{K_0}$	encryption
		$f(E_1, \cdots, E_n)$	function appl.

Table 1. Syntax of IoT-LySA.

In the syntax of systems, $\mathsf{0}$ denotes the null inactive system; a single node $\ell : [B]$ is uniquely identified by a label $\ell \in \mathcal{L}$ (which represents a node property such as location). Node components B are obtained by the parallel composition (through the operator $||$) of control processes P, and of a fixed number of sensors S. We assume that sensors are identified by an unique identifier $i \in \mathcal{I}_\ell$.

In the syntax of processes, $\mathsf{0}$ represents the *inactive* process (inactive components of a node are all coalesced). The process $\langle E_1, \cdots, E_k \rangle . P$ sends the tuple $E_1, \cdots, E_k$ to another process in the same node and then continues like P.

The process $\langle\!\langle E_1, \cdots, E_k \rangle\!\rangle \triangleright L.P$ sends the tuple $E_1, \ldots, E_k$ to the nodes whose labels are in L and evolves as P. The process $(E_1, \cdots, E_j;\ x_{j+1}, \cdots, x_k).P$ receives a tuple $E'_1, \cdots, E'_k$: if the first j terms of the received message pairwise match the first j terms of the input tuple, the message is accepted, otherwise is discarded (see later for details). The operator $\|$ describes parallel composition of processes, while $+$ denotes non-deterministic choice. An agent is a static definition of a parameterised process. Each agent identifier A has a unique defining equation of the form $A(y_1, \ldots, y_n) = P$, where $y_1, \ldots, y_n$ are distinct names occurring free in P. The process $\mathsf{decrypt}\ E\ \mathsf{as}\ \{E_1, \cdots, E_j; x_{j+1}, \cdots, x_k\}_{K_0}\ \mathsf{in}\ P$ tries to decrypt an encrypted value using the key K_0, provided that the first j elements of the decrypted term coincide with the terms E_j.

A sensor can perform an internal action τ or send an (encrypted) value v, gathered from the environment, to its controlling process and continues as S. We do not provide an explicit operation to read data from the environment but it can be easily implemented as an internal action.

Finally, in the syntax of term, a value represents a piece of data (e.g. keys or values read the environment). As said above, we have two kinds of disjoint variables: x are standard local variables, used as in π-calculus; while sensor variables z belong to a node and are globally accessible within it. As usual, we require that variables and names are disjoint. The encryption function $\{E_1, \cdots, E_k\}_{K_0}$ returns the result of encrypting values E_i for $i \in [1, k]$ with the key K_0. The term $f(E_1, \cdots, E_n)$ is the application of function f to n arguments; we assume given a set of primitive aggregation functions, e.g. functions for comparing values.

Working Example We set a simple IoT system up to keep the temperature under control inside a storehouse with perishable food (a big quadrangular room). We install four sensors, one for each corner of the storehouse. Each sensor S_i periodically senses (by means of the function $sense_i()$) the temperature and sends it through a wireless communication to a control unit P_c in the same node N_1. The unit P_c computes the average temperature and checks if it is within accepted bounds. If this is not the case, P_c sends an alarm through other nodes and the Cloud. We want to prevent an attacker from intercepting and manipulating data sent by sensors. A possible approach consists in exploiting the fact that sensors on the same side should sense the same temperature, with a difference that can be at most a given value ϵ. The control unit can easily detect anomalies and discard data tailored by an attacker, comparing values coming from the sensors that are on the same side of the room. But what happens if the attacker falsifies the data sent by more than one sensor? A possible solution consists in enabling some sensors (in our example S_1 and S_3) to use cryptography for obtaining reliable data. Nevertheless, before adopting this solution we would like to evaluate it by estimating its cost. The control process P_c in the node N_1 reads data from sensors, compares them and compute their average and sends them to the node N_2. The process Q_c of N_2 sends an `alarm` or an `ok` message to the node N_3 depending on the received data, together with the average temperature. The process R_c of N_3 is an Internet service that waits for messages from N_2 and

handles them (through the internal action τ). The IoT-LySA specification of the storehouse system follows:

$$
\begin{array}{lll}
N & = N_1 \mid N_2 \mid N_3 = \ell_1 : [P_c \parallel (S_0 \parallel S_1 \parallel S_2 \parallel S_3)] \mid \ell_2 : [Q_c \parallel \mathtt{0}] \mid \ell_3 : [R_c \parallel \mathtt{0}] \\
P_c & = (0;\ z_0).\,\tau.(\{1; z_1\}_{K_1}).\tau.(2;\ z_2).\,\tau.(\{3; z_3\}_{K_3}).\tau. \\
 & \quad \langle\!\langle cmp(z_0, ..., z_3), avg(z_0, ..., z_3) \rangle\!\rangle \triangleright \{\ell_2\}.\tau.P_c \\
Q_c & = (\mathtt{true}; x_{avg}).\langle\!\langle \mathtt{ok}, x_{avg} \rangle\!\rangle \triangleright \{\ell_3\}.\tau.Q_c + (\mathtt{false}; x_{avg}).\langle\!\langle \mathtt{alarm}, x_{avg} \rangle\!\rangle \triangleright \{\ell_3\}.\tau.Q_c \\
R_c & = (; w_{res}, w_{avg}).\tau.R_c \\
S_m & = \langle m, sense_m() \rangle.\,\tau.S_m \qquad m = 0, 2 \\
S_j & = \langle \{j, sense_j()\}_{K_j} \rangle.\,\tau.S_j \qquad j = 1, 3
\end{array}
$$

The function *cmp* performs consistency checks, by *comparing* data coming from insecure sensors with data from secure ones: it returns `true` if data are within the established bounds, `false` otherwise. The function *avg* computes the average of its arguments. We suppose that processes and sensors perform some internal activities (τ-actions). Another possible solution consists in having just one sensor that uses cryptography. This new system of nodes $\hat{N}$ differs from the previous one in the specification of the process $\hat{P}_c$ in the first node:

$$
\begin{array}{l}
\hat{P}_c = (0;\ z_0).\,\tau.(\{1; z_1\}_{K_1}).\tau.(2;\ z_2).\,\tau.(3;\ z_3).\,\tau. \\
\qquad \langle\!\langle halfcmp(z_0, z_1), avg(z_0, ..., z_3) \rangle\!\rangle \triangleright \{\ell_2\}.\tau.\hat{P}_c
\end{array}
$$

where the comparison function *halfcmp* uses only two arguments. We expect that this second solution is less expensive, and we apply our methodology to formally compare the relative costs of the two solutions.

Enhanced Operational Semantics. To estimate cost, we give an *enhanced* reduction semantics following [2,6,7]. The underlying idea is that each transition is enriched with an *enhanced label* θ, which records information about the transition. Actually, we label transitions for communications and decryptions. For communications, we record the action (input or output) with the corresponding prefixes, and the labels of the involved nodes. For decryption, we store the label of the node performing the operation and information about the data. Note that in the following we use the abbreviations *out*, *in*, *dec*, for denoting the communication prefixes, the decryption constructs and the possible function calls f inside them. We can obtain a standard semantics by simply removing the labels.

Definition 1. *Given* $\ell, \ell_O, \ell_I, \ell_D \in \mathcal{L}$, enhanced labels *theta are defined as:*

$$
\begin{array}{lll}
\Theta \ni \theta ::= & \langle \ell\ \{out\}, \ell\ \{in\} \rangle & \textit{internal secure communication} \\
 & \langle \ell\ out, \ell\ in \rangle & \textit{internal communication} \\
 & \langle \ell_O\ out, \ell_I\ in \rangle & \textit{inter-nodes communication} \\
 & \{\ell_D\ dec\} & \textit{decryption of a message}
\end{array}
$$

As usual, our semantics consists of the standard structural congruence $\equiv$ on nodes, processes and sensors and of a set of rules defining the transition relation. Our notion of *structural congruence* $\equiv$ is standard except for the following congruence rule for processes that equates a multi-output with empty set of receivers to the inactive process: $\langle\!\langle E_1, \cdots, E_k \rangle\!\rangle \triangleright \emptyset.\mathsf{0} \equiv \mathsf{0}$.

Our *reduction relation* $\xrightarrow{\theta} \subseteq \mathcal{N} \times \mathcal{N}$ is defined as the least relation on closed nodes, processes and sensors that satisfies a set of inference rules. Our rules are quite standard apart from the five rules for communications shown in Tab. 2 and briefly commented below. We assume the standard meaning for terms $[\![E]\!]$.

(Sensor-Com)

$$\ell : [\langle i, v_i \rangle.\, S_i \parallel (i;\ z_i).\, P \parallel B] \xrightarrow{\langle \ell\ out, \ell\ in \rangle} \ell : [S_i \parallel P\{v_i/z_i\} \parallel B\{v_i/z_i\}]$$

(Crypto-Sensor-Com)

$$\ell : [\langle \{i, v_i\}_K \rangle.\, S_i \parallel (\{i; z_i\}_K).P \parallel B] \xrightarrow{\langle \ell\ \{out\}, \ell\ \{in\} \rangle} \ell : [S_i \parallel P\{v_i/z_i\} \parallel B\{v_i/z_i\}]$$

(Intra-Com)

$$\frac{\bigwedge_{i=1}^{k} v_i = [\![E_i]\!] \ \wedge\ \bigwedge_{i=1}^{j} [\![E_i]\!] = [\![E'_i]\!]}{\ell : [\langle E_1, ..., E_k \rangle.\, P \parallel (E'_1, ..., E'_j; x_{j+1}, ..., x_k).Q \parallel B] \xrightarrow[\langle \ell\ out, \ell\ in \rangle]{} \ell : [P \parallel Q\{v_{j+1}/x_{j+1}, ..., v_k/x_k\} \parallel B]}$$

(Multi-Com)

$$\frac{\ell_2 \in L \ \wedge\ Comp(\ell_1, \ell_2) \ \wedge\ \bigwedge_{i=1}^{k} v_i = [\![E_i]\!] \ \wedge\ \bigwedge_{i=1}^{j} [\![E_i]\!] = [\![E'_i]\!]}{\ell_1 : [\langle\!\langle E_1, ..., E_k \rangle\!\rangle \triangleright L.P_{11} \parallel B_P] \mid \ell_2 : [(E'_1, ..., E'_j; x_{j+1}, ..., x_k).Q_{11} \parallel B_Q] \xrightarrow[\langle \ell_1\ out, \ell_2\ in \rangle]{} \ell_1 : [\langle\!\langle E_1, ..., E_k \rangle\!\rangle \triangleright L \setminus \{\ell_2\}.P_{11} \parallel B_P] \mid \ell_2 : [Q_{11}\{v_{j+1}/x_{j+1}, ..., v_k/x_k\} \parallel B_Q]}$$

Table 2. Operational semantic rules for communication.

The rule (*Sens-Com*) is for communications among sensors and processes: the variables used in the input are assumed *global* inside the node, in such a way that sensors are considered as a shared data structure $z_1, \cdots, z_n$. Therefore, the substitution is performed in all the processes of the node. The rule (*Crypto-Sens-Com*) is similar but it also requires that the receiving process successfully decrypts the encrypted data sent by a sensor. The rule (*Intra-Com*) is for intra-node communications. This construct implements also a matching feature: the communication succeeds, as long as the first j values of the message match the evaluations of the first j terms in the input. If this is the case, the result of evaluating each E_i is bound to each x_i.

The rule (*Multi-Com*) implements the inter nodes communication: the communication between the node labelled ℓ_1 and the node ℓ_2 succeeds, provided that (i) ℓ_2 is in the set L of receivers, (ii) the two nodes are compatible according to the compatibility function $Comp$, and (iii) the matching mechanism succeeds. If this is the case, the sender removes ℓ' from the set of receivers L, while in the second node, the receiving process continues bounding the result of each E_i to each variable x_i. Outputs terminate when all the nodes in L have received the message (see the congruence rule). Note that point-to-point communication amounts to

the case in which L is a singleton. The compatibility function $Comp$ defined over node labels is used to model constraints on communication, e.g. proximity, with $Comp(\ell_1, \ell_2)$ that yields true only when the two nodes ℓ_1, ℓ_2 are in the same transmission range. Of course, this function could be enriched for considering other notions of compatibility.

Hereafter, we assume the standard notion of transition system. Intuitively, it is a graph, in which systems of nodes form the nodes and (labelled) arcs represent the possible transitions between them. As will be clearer in the next section, we will only consider *finite* state systems, because finite states have an easier stochastic analysis. Note that this does not mean that the behaviour of such processes is finite, because their transition systems may have loops.

Example (cont'd) Back to our example, consider the following run of the first system where, for brevity, we ignored their internal actions τ

$$N \xrightarrow{\theta_0} N' \xrightarrow{\theta_1} N'' \xrightarrow{\theta_2} N''' \xrightarrow{\theta_3} N'''' \xrightarrow{\theta_{4i}} \begin{cases} N_0''''' \xrightarrow{\theta_{50}} N \text{ if } i = 0 \\ N_1''''' \xrightarrow{\theta_{51}} N \text{ if } i = 1 \end{cases}$$

the systems of nodes $N', N'', N''', N'''', N'''''$ are the intermediate ones reached from N during the computation and the labels θ_j annotate the j^{th} transition (θ_{ji} depending on the branch of the summation). In the run, fully specified below, the sensors of N_1 send a message to the process P_c, which checks the received data and sends the checking result to N_2. We denote with P_c' (Q_c', R_c', resp.) the continuations of P_c (Q_c, R_c, resp.) after the first input prefixes, with v_{comp} the value $cmp(v_0, ..., v_3)$, with v_{avg} the value $avg(v_0, ..., v_3)$, and with v_{res_i} (with $i = 0, 1$) the value `ok` (`alarm` respectively), depending on which branch of the summation is chosen. The evolution of the second system $\hat{N}$ is analogous to the one of N: the transition labels are such that $\hat{\theta}_{4i} = \langle \ell_1 \langle \langle halfcmp(v_0, v_1), avg(v_0, \cdots, v_3) \rangle \rangle, \ell_2(v_{bool}; x_{avg}) \rangle$ and $\hat{\theta}_l = \theta_l$ for $l \neq 4i$ (the transition labels θ_l are presented below, after the run of N).

$$\begin{aligned}
N \quad &= \ell_1 : [(0;\ z_0).\, P_c' \parallel P \parallel (\langle 0, sense_0() \rangle.\, \tau.S_0 \parallel S_1 \parallel S_2 \parallel S_3)] \mid N_2 \mid N_3 \xrightarrow{\theta_0} \\
N' \quad &= \ell_1 : [P_c'\{0/z_0\} \parallel (\tau.S_0 \parallel S_1 \parallel S_2 \parallel S_3)] \mid N_2 \mid N_3 \xrightarrow{\theta_1}\xrightarrow{\theta_2}\xrightarrow{\theta_3} \\
N'''' \quad &= \ell_1 : [P_c'\{0/z_0, 1/z_1, 2/z_2, 3/z_3\} \parallel (S_0 \parallel S_1 \parallel S_2 \parallel S_3)] \mid N_2 \mid N_3 = \\
&\quad \ell_1 : [\langle\!\langle v_{comp}, v_{avg} \rangle\!\rangle \triangleright \{\ell_2\}.\tau.P_c \parallel (S_0 \parallel S_1 \parallel S_2 \parallel S_3)] \mid \\
&\quad \ell_2 : [(\texttt{true}; x_{avg}).\langle\!\langle \texttt{ok}, x_{avg} \rangle\!\rangle \triangleright \{\ell_3\}.\tau.Q_c + \\
&\qquad (\texttt{false}; x_{avg}).\langle\!\langle \texttt{alarm}, x_{avg} \rangle\!\rangle \triangleright \{\ell_3\}.\tau.Q_c] \mid N_3 \xrightarrow{\theta_{4i}} \\
N_i''''' \quad &= \ell_1 : [P_c \parallel P \parallel (S_0 \parallel S_1 \parallel S_2 \parallel S_3)] \mid \\
&\quad \ell_2 : [Q_c'\{v_{avg}/x_{avg}\}) \parallel 0] \mid \ell_3 : [(; w_{res}, w_{avg}).\tau.R_c \parallel 0] = \\
&\quad \ell_1 : [P_c \parallel P \parallel (S_0 \parallel S_1 \parallel S_2 \parallel S_3)] \mid \\
&\quad \ell_2 : [\langle\!\langle v_{res_i}, v_{avg} \rangle\!\rangle \triangleright \{\ell_3\}.\tau.Q_c \parallel 0] \mid \ell_3 : [(; w_{res}, w_{avg}).\tau.R_c \parallel 0] \xrightarrow{\theta_{5i}} \\
N \quad &= \ell_1 : [P_c \parallel P \parallel (S_0 \parallel S_1 \parallel S_2 \parallel S_3)] \mid \ell_2 : [Q_c \parallel 0] \mid \ell_3 : [R_c'\{v_{res_i}/w_{res}, v_{avg}/w_{avg}\} \parallel 0]
\end{aligned}$$

$$\begin{aligned}
&\theta_0 = \theta_2 = \langle \ell_1 \langle j, v_j \rangle, \ell_1(j; z_i) \rangle \\
&\theta_1 = \theta_3 = \langle \ell_1 \langle \{j, v_j\}_{K_i} \rangle, \ell_1(\{j; z_j\}_{K_i}) \rangle \\
&\theta_{4i} = \langle \ell_1 \langle \langle cmp(v_0, \cdots, v_3), avg(v_0, \cdots, v_3) \rangle \rangle, \ell_2(v_{bool}; x_{avg}) \rangle \\
&\theta_{5i} = \langle \ell_2\ \langle \langle v_{res_i}, v_{avg} \rangle \rangle, \ell_3(w_{res}, w_{avg}) \rangle
\end{aligned}$$

3 Stochastic Semantics

We now show how to generate a Continuous Time Markov Chains (CTMC) from a transition system (see [17] for more details). First, we introduce functions over the enhanced labels to associate costs to transitions. Here, the cost of a system is specified in term of the *time* spent for transitions, and it depends on the performed action as well as on the involved nodes. However, we can easily treat other quantitative properties, e.g. energy consumption. Intuitively, cost functions define exponential distributions, from which we compute the rates at which a system evolves and the corresponding CTMC. Then, to evaluate the performance we calculate the stationary distribution of the CTMC and the transition rewards.

Cost Functions. Our cost functions assign a *rate* to each transition with label ϑ. To define this rate, we suppose to execute each action on a dedicated architecture that only performs that action, and we estimate the corresponding duration. To model the performance degradation due to the run-time support, we introduce a scaling factor for r for each routine called by the implementation under consideration. Here, we just propose a format for these functions, with parameters that depend on the nodes to be instantiated on need. For instance, in a node where the cryptographic operations are performed at very high speed (e.g. by a cryptographic accelerator), but with a slow link (low bandwidth), the time will be low for encryptions and high for communication; vice versa, in a node offering a high bandwidth, but poor cryptography resources.

Technically, we interpret costs as parameters of exponential distributions $F(t) = 1 - e^{-rt}$, with *rate* r and t as time parameter (general distributions are also possible see [18]): the transition rate r is the parameter that identifies the exponential distribution of the duration times of the transition, as usual in stochastic process algebras (e.g. [8]). The shape of $F(t)$ is a curve that grows from 0 asymptotically approaching 1 for positive values of its argument t. The parameter r determines the slope of the curve: the greater r, the faster $F(t)$ approaches its asymptotic value. The exponential distributions that we use enjoy the *memoryless property*, i.e. the occurrence of a new transition does not depend on the previous ones. We also assume that transitions are *time homogeneous* (the transitions enabled in a given state cannot be disabled by the flow of time).

We define in a few steps the function that associates rates with the (enhanced labels of) communication and decryption transitions. For the sake of simplicity, we ignore the costs for other primitives, e.g. constant invocation, parallel composition, summation, τ actions (see [17] for a complete treatment); we further neglect the sensor cost of sensing from the environment We need the auxiliary function $f_E : \mathcal{E} \to \mathbb{R}^+$ that estimates the effort needed to manipulate terms.

- $f_E(v) = size(v)$
- $f_E(\{E_1, \ldots, E_k\}_{K_0}) = f_{enc}(f_E(E_1), ..., f_E(E_1), crypto\_system, kind(K_0))$

The size of a value v matters. For an encrypted term, we use the function f_{enc}, which depends on the terms to encrypt, on the used crypto-system and on the

kind (short/long, short-term/long-term) of the key. The function $\$_\alpha : \mathcal{A} \to \mathbb{R}^+$ assigns costs to I/O and decryption prefix actions $\alpha \in \mathcal{A}$.

- $\$_\alpha(\langle E_1, \ldots, E_k \rangle) = f_{out}(f_E(E_1), ..., f_E(E_1), bw)$
- $\$_\alpha((E_1, \ldots, E_j; x_{j+1}, \ldots, x_k)) = f_{in}(f_E(E_1), ..., f_E(E_j), match(j), bw)$
- $\$_\alpha(\mathsf{decrypt}\ E\ \mathsf{as}\ \{E_1, \cdots, E_j; x_{j+1}, \cdots, x_k\}_{E_0}) =$
  $f_{dec}(f_E(E), crypto\_system, kind(K_0), match(j))$

the send and receive primitives. Besides the implementation cost due to their own algorithms, the functions above depend on the bandwidth of the channel (represented by bw), on the cost of the exchanged terms (computed by f_E), and on the nodes involved in the communication. Moreover, the inter-node communication depends on the proximity-relation (e.g. the transmission range between nodes), represented here by the function $f_{<>}(\ell_O, \ell_I)$. Also, the cost of an input depends on the number of required matchings (represented by $match(j)$). Finally, the function f_{dec} represents the cost of a decryption, whose cost is similar to the one for encryption, with the additional cost of matchings. Finally, the function $\$: \Theta \to \mathbb{R}^+$ associates rates with enhanced labels.

- $\$(\langle \ell_O\ out, \ell_I\ in \rangle) = f_{<>}(\ell_O, \ell_I) \cdot \min\{\$_\alpha(out, \ell_O), \$_\alpha(in, \ell_I)\}$
- $\$\langle \ell\ dec \rangle = \$_\alpha(dec, \ell)$

As mentioned above, the two partners independently perform some low-level operations locally to their nodes, labelled ℓ_O and ℓ_I. Each label leads to a delay in the rate of the corresponding action. Thus, the cost of the slower partner corresponds to the minimum cost of the operations performed by the participants in isolation. Indeed the lower the cost, i.e. the rate, the greater the time needed to complete an action and hence the slower the speed of the transition (and the slower $F(t) = 1 - e^{-rt}$ approaches its asymptotic value).

Note that we do not fix the actual cost function: we only propose for it a set of parameters to reflect some features of an idealised architecture. Although very abstract, this suffices to make our point. A precise instantiation comes with the refinement steps from specification to implementations as soon as actual parameters become available.

Example (cont'd) We now associate a rate to each transition in the transition system of the system of nodes N, just N for brevity. To illustrate our methodology, we assume that the coefficients due to the nodes amount to 1. We instantiate the cost functions given above, by using the following parameters in the denominator: (*i*) `e` and `d` for encrypting and for decrypting; (*ii*) `s` and `r` for sending and for receiving; (*iii*) `m` for pattern matching; and (*iv*) `f` for the application of the aggregate function f, whose cost is proportional to the number of its arguments.

- $f_E(a) = 1$
- $f_E(\{E_1, \ldots, E_k\}_{K_0}) = \frac{\mathtt{e}}{\mathtt{s}} \cdot \sum_{i=1}^{k} f_E(E_i) + 1$
- $\$_\alpha(\langle E_1, \ldots, E_k \rangle) = \frac{1}{\mathtt{s} \cdot \sum_{i=1}^{i} f_E(E_i)}$
- $\$_\alpha((E_1, \ldots, E_j; x_{j+1}, \ldots, x_k)) = \frac{1}{\mathtt{r} \cdot k + \mathtt{m} \cdot j}$
- $\$_\alpha(\mathsf{decrypt}\ E\ \mathsf{as}\ \{E_1, \cdots, E_j; x_{j+1}, \cdots, x_k\}_{K_0}) = \frac{1}{\mathtt{d} \cdot k + \mathtt{m} \cdot j}$
- $\$_\alpha(f(E_1, \cdots, E_k)) = \frac{1}{\mathtt{f} \cdot k}$

These parameters represent the time spent performing the corresponding action on a single term. Intuitively, the greater the time duration is, the smaller the rate. Since transmission is usually more time-consuming than reception, the rate of a communication is that of output. The rates of the transitions of N and $\hat{N}$ are $c_j = \$(\theta_j)$ and $\hat{c}_j = \$(\hat{\theta}_j)$, and $c_{ji} = \$(\theta_{ji})$ and $\hat{c}_{ji} = \$(\hat{\theta}_{ji})$ $(j \in [4,5],\ i \in [0,1])$.

$$c_0 = c_2 = \tfrac{1}{2\mathtt{s}},\quad c_1 = c_3 = \tfrac{1}{2\mathtt{e}+\mathtt{s}},\quad c_{4i} = \tfrac{1}{8\mathtt{f}+2\mathtt{s}}\quad c_{5i} = \tfrac{1}{\mathtt{s}}$$
$$\hat{c}_0 = \hat{c}_2 = \hat{c}_3 = \tfrac{1}{2\mathtt{s}},\quad \hat{c}_1 = \tfrac{1}{2\mathtt{e}+\mathtt{s}},\quad \hat{c}_{4i} = \tfrac{1}{6\mathtt{f}+2\mathtt{s}}\quad \hat{c}_{5i} = \tfrac{1}{\mathtt{s}}$$

For instance, the rate of the second transition is: $c_1 = \$(\theta_1) = \frac{1}{2\mathtt{e}+\mathtt{s}}$, where $\frac{1}{2\mathtt{e}+\mathtt{s}} = min\{\frac{1}{2\mathtt{e}+\mathtt{s}}, \frac{1}{2\mathtt{d}+\mathtt{r}+\mathtt{m}}\}$. Note that our costs can be further refined; we could e.g. make the transmission rate also depend on the distance between the nodes, when non internal to a node.

Stochastic Analysis Now, we transform the transition system N into its corresponding $CTMC(N)$, by using the above rates. Afterwards, we can calculate the actual performance measures, e.g. the throughput or utilisation of a certain resource (see [16] for more details on the theory of stochastic processes).

Actually, the *transition rate* $q(N_i, N_j)$ at which a system jumps from N_i to N_j is the sum of the single rates ϑ_k of all the possible transitions from N_i to N_j. Given a transition system N, the corresponding CTMC has a state for each node in N, and the arcs between states are obtained by coalescing all the arcs with the same source and target in N. Recall that a CTMC can be seen as a directed graph and that its matrix $\mathbf{Q}$, the *generator matrix*, (apart from its diagonal) represents its adjacency matrix. Note that $q(N_i, N_j)$ coincides with the off-diagonal element q_{ij} of the generator matrix $\mathbf{Q}$. Hence, hereafter we will use indistinguishably CTMC and its corresponding $\mathbf{Q}$ to denote a Markov chain. More formally, the entries of $\mathbf{Q}$ are defined as follows.

$$q_{ij} = \begin{cases} q(N_i, N_j) = \sum\limits_{\theta_k} \$(\theta_k) \text{ if } i \neq j \ \wedge\ N_i \xrightarrow{\theta_k} N_j \\ -\sum\limits_{j=0, j\neq i}^{n} q_{ij} \qquad\qquad \text{if } i = j \end{cases}$$

Performance measures are usually obtained by computing the stationary distributions of $CTMCs$, since they should be taken over long periods of time to be significant. The *stationary probability distribution* of a CTMC is $\Pi = (X_0, \ldots, X_{n-1})$ s.t. $\Pi^T \mathbf{Q} = \mathbf{0}$ and $\sum_{i=0}^{n} X_i = 1$, which *uniquely* exists if the transition system is finite and has a cyclic initial state.

Example (cont'd) Consider the transition system corresponding to N that is, as required above, finite and with a cyclic initial state. We derive the following generator matrix $\mathbf{Q_1}$ of $CTMC(N)$ and the corresponding stationary distribution $\mathit{\Pi}_1$, where $C = 4\mathtt{s} + 2\mathtt{e} + 2\mathtt{f}$, by solving the system of linear equations $\mathit{\Pi}_1^T \mathbf{Q}_1 = \mathbf{0}$ and $\sum_{i=0}^{n} X_i = 1$, where $\mathit{\Pi}_1 = (X_0, \cdots, X_6)$. Similarly, we can derive the generator matrix $\hat{Q}_1'$ and the corresponding stationary distribution $\hat{\mathit{\Pi}}_1$ for the transition system corresponding to $\hat{N}$, where $\hat{C} = 9\mathtt{s} + 2\mathtt{e} + 3\mathtt{f}$.

$$\mathbf{Q_1} = \begin{bmatrix} -c_0 & c_0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -c_1 & c_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -c_2 & c_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -c_3 & c_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & -(c_{40} + c_{41}) & c_{40} & c_{41} \\ c_{50} & 0 & 0 & 0 & 0 & -c_{50} & 0 \\ c_{51} & 0 & 0 & 0 & 0 & 0 & -c_{51} \end{bmatrix}$$

$$\mathit{\Pi}_1 = \left[\tfrac{\mathtt{s}}{\mathtt{C}},\ \tfrac{(2\mathtt{e}+\mathtt{s})}{2\mathtt{C}},\ \tfrac{\mathtt{s}}{2\mathtt{C}},\ \tfrac{(2\mathtt{e}+\mathtt{s})}{\mathtt{C}},\ \tfrac{(4\mathtt{f}+\mathtt{s})}{2\mathtt{C}},\ \tfrac{\mathtt{s}}{4\mathtt{C}},\ \tfrac{\mathtt{s}}{4\mathtt{C}} \right]$$

$$\hat{\mathit{\Pi}}_1 = \left[\tfrac{2\mathtt{s}}{\hat{\mathtt{C}}},\ \tfrac{(2\mathtt{e}+\mathtt{s})}{\hat{\mathtt{C}}},\ \tfrac{2\mathtt{s}}{\hat{\mathtt{C}}},\ \tfrac{2\mathtt{s}}{\hat{\mathtt{C}}},\ \tfrac{(3\mathtt{f}+\mathtt{s})}{\hat{\mathtt{C}}},\ \tfrac{\mathtt{s}}{2\hat{\mathtt{C}}},\ \tfrac{\mathtt{s}}{2\hat{\mathtt{C}}} \right]$$

To define performance measures for a system N, we define the corresponding *reward structure*, following [9,8]. Usually, a reward structure is a function that associates a reward with any state passed through in a computation of N. We compute rewards from rates of transitions [17]. To measure the throughput of a system, i.e. the amount of useful work accomplished per unit time, a reasonable choice is to use as nonzero reward a value equal to the rate of the corresponding transition. The reward structure of a system N is a vector of rewards, whose size amounts to the number of N states. By looking at the stationary distribution and varying the reward structure, we can compute different performance measures. The *total reward* is obtained by summing the values of the stationary distribution $\mathit{\Pi}$ multiplied by the corresponding reward structure ρ.

Definition 2. *Given a system N, let $\mathit{\Pi} = (X_0, \ldots, X_{n-1})$ be its stationary distribution and $\rho = \rho(0), ..., \rho(n-1)$ be its reward structure. The* total reward *of N is computed as $R(N) = \sum_i \rho(i) \cdot X_i$.*

Example (cont'd) To evaluate the relative efficiency of the two systems of nodes, we compare the throughput of both, i.e. the number of instructions executed per time unit. The throughput for a given activity is found by first associating a transition reward equal to the activity rate with each transition. In our systems each transition is fired only once. Also, the graph of the corresponding CTMC is cyclic and all the labels represent different activities. Therefore the throughput of all the activities is the same, and we can freely choose one of them to compute the throughput of N. Thus we associate a transition reward equal to its rate with the last communication and a null transition reward with all the others communications. The total reward $R(N)$ of the system then amounts to $\frac{1}{2(8\mathtt{s}+4\mathtt{e}+4\mathtt{f})}$, while $R(\hat{N})$ to $\frac{1}{2(9\mathtt{s}+2\mathtt{e}+3\mathtt{f})}$. By comparing the two throughputs, it is straightforward to

obtain that $R(N) < R(\hat{N})$, i.e. that, as expected, $\hat{N}$ performs better. To use this measure, it is necessary to instantiate our parameters under various hypotheses, depending on several factors, such as the network load, the packet size, and so on. Furthermore, we need to consider the costs of cryptographic algorithms and how changing their parameters impact on energy consumption and on the guaranteed security level (see e.g. [15]).

4 Conclusions

In the IoT scenario security is critical but it is hard to address in an affordable way due to the limited computational capabilities of smart objects. We have presented the first steps towards a formal framework that supports designers in specifying an IoT system and in estimating the cost of security mechanisms. A key feature of our approach is that quantitative aspects are symbolically represented by parameters. Actual values are obtained as soon as the designer provides some additional information about the hardware and the network architecture and the cryptographic algorithms relative to the system in hand. By abstractly reasoning about these parameters designers can compare different implementations of the same IoT system, and choose the one that ensures the best trade-off between security guarantees and their price. In practice, we considered a subset of the process algebra IoT-LySA [4] and we adapted the technique in [1] to determine the costs of using/not using cryptographic measures in communications and to reason about the cost-security trade-offs. In particular, we defined an enhanced semantics, where each system transition is associated with a rate in the style of [7,17]. From the rates we derive a CTMC, through which we could perform cost evaluation, by using standard techniques and tools [19,22]. As future work, we plan to assess our proposal considering a more complete case study and considering different metrics as time, network bandwidth and energy consumption.

Our approach follows the well-established line of research about performance evaluation through process calculi and probabilistic model checking (see [10,11] for a survey). To the best of our knowledge, the application of formal methods to IoT systems or to wireless or sensor networks have not been largely studied and only a limited number of papers in the literature addressed the problem from a process algebras perspective, e.g. [13,12,14,21], to cite only a few. In [5] the problem of modelling and estimating the communication cost in an IoT scenario is tackled through Stochastic Petri Net. Their approach is similar to ours: they derive a CTMC from a Petri Net describing the system and proceed with the performance evaluation by using standard tools. Differently from us, they focus not on the cost of security but only on the one of communication (they do not use cryptographic primitives). In [20] a performance comparison between the security protocols IPSec and DTLS is presented, in particular by considering their impact on the resources of IoT devices with limited computational capabilities. They modified protocols implementations to make them properly run on the devices.

An extensive experimental evaluation study on these protocols shows that both their implementations ensure a good level of end-to-end security.

References

1. Bodei, C., Buchholtz, M., Curti, M., Degano, P., Nielson, F., Nielson, H.R., Priami, C.: On evaluating the performance of security protocols. In: Proc. of PaCT'05. LNCS, vol. 3606, pp. 1 – 15. Springer (2005)
2. Bodei, C., Buchholtz, M., Degano, P., Nielson, F., Nielson, H.R.: Static validation of security protocols. Journal of Computer Security 13(3), 347–390 (2005)
3. Bodei, C., Degano, P., Ferrari, G.L., Galletta, L.: A step towards checking security in IoT. In: Procs. of ICE 2016. EPTCS, vol. 223, pp. 128–142 (2016)
4. Bodei, C., Degano, P., Ferrari, G.L., Galletta, L.: Where do your IoT ingredients come from? In: Procs. of Coordination 2016. LNCS, vol. 9686. Springer (2016)
5. Chen, L., Shi, L., Tan, W.: Modeling and performance evaluation of internet of things based on petri nets and behavior expression. Research Journal of Applied Sciences, Engineering and Technology 4(18), 3381–3385 (2012)
6. Degano, P., Priami, C.: Non interleaving semantics for mobile processes. Theoretical Computer Science 216 (1999)
7. Degano, P., Priami, C.: Enhanced operational semantics. ACM Computing Surveys 33(2), 135 – 176 (July 2001)
8. Hillston, J.: A Compositional Approach to Performance Modelling. Cambridge University Press (1996)
9. Howard., R.: Dynamic Probabilistic Systems: Semi-Markov and Decision Systems, vol. Volume II. Wiley (1971)
10. Kwiatkowska, M., Norman, G., Parker, D.: Stochastic model checking. In: Procs. of Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07). vol. LNCS 4486, pp. 220–270 (2007)
11. Kwiatkowska, M., Parker, D.: Advances in probabilistic model checking. In: Procs. of Software Safety and Security - Tools for Analysis and Verification. vol. 33, pp. 126–151. IOS Press (2012)
12. Lanese, I., Bedogni, L., Felice, M.D.: Internet of things: a process calculus approach. In: Procs of Symp. on Applied Computing (SAC '13). pp. 1339–1346. ACM (2013)
13. Lanese, I., Sangiorgi, D.: An operational semantics for a calculus for wireless systems. Theor. Comput. Sci. 411(19), 1928–1948 (2010)
14. Lanotte, R., Merro, M.: A semantic theory of the Internet of Things. In: Procs. of Coordination 2016. LNCS, vol. 9686, pp. 157–174. Springer (2016)
15. Lee, J., Kapitanova, K., Son, S.: The price of security in wireless sensor networks. Computer Networks 54(17), 2967–2978 (2010)
16. Nelson, R.: Probability, Stochastic Processes and Queeing Theory. Springer (1995)
17. Nottegar, C., Priami, C., Degano, P.: Performance evaluation of mobile processes via abstract machines. Transactions on Software Engineering 27(10) (2001)
18. Priami, C.: Language-based performance prediction of distributed and mobile systems. Information and Computation 175, 119–145 (2002)
19. Reibnam, A., Smith, R., Trivedi, K.: Markov and Markov reward model transient analysis: an overview of numerical approaches. European Journal of Operations Research (40), 257–267 (1989)
20. Rubertis, A.D., Mainetti, L., Mighali, V., Patrono, L., Sergi, I., Stefanizzi, M., Pascali, S.: Performance evaluation of end-to-end security protocols in an Internet of Things. In: Proc. of (SoftCOM). pp. 1–6. IEEE (2013)

21. Singh, A., Ramakrishnan, C.R., Smolka, S.: A process calculus for mobile ad hoc networks. Sci. Comput. Program. 75(6), 440–469 (2010)
22. Stewart, W.J.: Introduction to the numerical solutions of Markov chains. Princeton University Press (1994)