Secure Apps in the Fog: Anything to Declare?

Antonio Brogi, Gian-Luigi Ferrari, and Stefano Forti

Department of Computer Science,
University of Pisa, Italy
name.surname@di.unipi.it

Abstract. Assessing security of application deployments in the Fog is a non-trivial task, having to deal with highly heterogeneous infrastructures containing many resource-constrained devices. In this paper, we introduce: *(i)* a declarative way of specifying security capabilities of Fog infrastructures and security requirements of Fog applications, and *(ii)* a (probabilistic) reasoning strategy to determine application deployments and to quantitatively assess their security level, considering the trust degree of application operators in different Cloud/Fog providers. A lifelike example is used to showcase a first proof-of-concept implementation and to illustrate how it can be used in synergy with other predictive tools to optimise the deployment of Fog applications.

Keywords: Fog computing · Application Deployment · Security Assessment · Executable Specifications · Probabilistic Logic Programming · Trust.

1 Introduction

Fog computing [9] aims at better supporting the growing processing demand of (time-sensitive and bandwidth hungry) Internet of Things (IoT) applications by selectively pushing computation closer to where data is produced and exploiting a geographically distributed multitude of heterogeneous devices (e.g., personal devices, gateways, micro-data centres, embedded servers) spanning the continuum from the Cloud to the IoT. As a complement and an extension of the Cloud, the Fog will naturally share with it many security threats and it will also add its peculiar ones. On the one hand, Fog computing will increase the number of security enforcement points by allowing local processing of private data closer to the IoT sources. On the other hand, the Fog will be exposed to brand new threats for what concerns the trust and the physical vulnerability of devices. In particular, Fog deployments will span various service providers - some of which may be not fully trustable - and will include accessible devices that can be easily hacked, stolen or broken by malicious users [25]. Security will, therefore, play a crucial role in the success of the Fog paradigm and it represents a concern that should be addressed *by-design* at all architectural levels [26, 37]. The Fog calls for novel technologies, methodologies and models to guarantee adequate security (privacy and trust) levels to Fog deployments even when relying upon resource-constrained devices [8].

Meanwhile, modern computing systems are more and more made from distributed components – such as in service-oriented and micro-service based architectures – what makes it challenging to determine how they can be *best-placed* so to fulfil various application requirements. In our previous work, we proposed a model and algorithms to determine eligible deployments of IoT applications to Fog infrastructures [4] based on hardware, software and QoS requirements. Our prototype – FogTorchΠ – implements those algorithms and permits to estimate the QoS-assurance, the resource consumption in the Fog layer [5] and the monthly deployment cost [6] of the output eligible deployments. Various other works tackled the problem of determining "optimal" placements of application components in Fog scenarios, however, none included a quantitative security assessment to holistically predict security guarantees of the deployed applications, whilst determining eligible application deployments. Therefore, there is a clear need to evaluate *a priori* whether an application will have its security requirements fulfilled by the (Cloud and Fog) nodes chosen for the deployment of its components. Furthermore, due to the mission-critical nature of many Fog applications (e.g., e-health, disaster recovery), it is important that the techniques employed to reason on security properties of deployed multi-component applications are configurable and well-founded.

In this paper, we propose a methodology (SecFog) to (quantitatively) assess the security level of multi-component application deployments in Fog scenarios. Such quantitative assessment can be used both alone – to maximise the security level of application deployments – and synergically with other techniques so to perform multi-criteria optimisations and to determine the *best* placement of application components in Fog infrastructure. This work allows application deployers to specify security constraints both at the level of the components and at the level of the application as a whole. As per recent proposals in the field of AI [3], it exploits probabilistic reasoning to account for reliability and trust, whilst capturing the uncertainty typical of in Fog scenarios. Therefore, we propose: *(i)* a declarative methodology that enables writing an executable specification of the security policies related to an application deployment to be checked against the security offerings of a Fog infrastructure, *(ii)* a reasoning methodology that can be used to look for secure application deployments and to assess the security levels guaranteed by any input deployment, and *(iii)* a first proof-of-concept implementation of SecFog which can be used to optimise security aspects of Fog application deployments along with other metrics.

The rest of this paper is organised as follows. After reviewing some related work (Section 2), we offer an overview of SecFog and we introduce a motivating example (Section 3). Then, we present our proof-of-concept implementation of SecFog and we show how it can be used to determine application deployment whilst maximising their security level (Section 4). Finally, we show how SecFog can be used with FogTorchΠ to identify suitable trade-offs among QoS-assurance, resource usage, monthly cost and security level of eligible deployments (Section 2), and we briefly conclude with some directions for future work (Section 6).

2 Related Work

Among the works that studied the placement of multi-component applications to Cloud nodes, very few approaches considered security aspects when determining eligible application deployments, mainly focussing on improving performance, resource usage and deployment cost [18, 21], or on performing identification of potential data integrity violations based on pre-defined risk patterns [28]. Indeed, existing research considered security mainly when treating the deployment of business processes to (federated) multi-Clouds (e.g., [23, 12, 36]). Similar to our work, Luna et al. [19] were among the first to propose a quantitative reasoning methodology to rank single Cloud providers based on their security SLAs, and with respect to a specific set of (user-weighted) security requirements. Recently, swarm intelligence techniques [21] have been exploited to determine eligible deployments of composite Cloud applications, considering a risk assessment score based on node vulnerabilities.
Fog computing introduces new challenges, mainly due to its pervasive geo-distribution and heterogeneity, need for QoS-awareness, dynamicity and support to interactions with the IoT, that were not thoroughly studied in previous works addressing the problem of application deployment to the Cloud [32, 35]. Among the first proposals investigating these new lines, [15] proposed a Fog-to-Cloud search algorithm as a first way to determine an eligible deployment of (multi-component) DAG applications to tree-like Fog infrastructures. Their placement algorithm attempts the placement of components *Fog-to-Cloud* by considering hardware capacity only. An open-source simulator – iFogSim – has been released to test the proposed policy against Cloud-only deployments. Building on top of iFogSim, [20] refines tries to guarantee the application service delivery deadlines and to optimise Fog resource exploitation. Also [33] used iFogSim to implement an algorithm for optimal online placement of application components, with respect to load balancing. Recently, exploiting iFogSim, [13] proposed a distributed search strategy to find the best service placement in the Fog, which minimises the distance between the clients and the most requested services, based on request rates and available free resources. [17, 30] proposed (linearithmic) heuristic algorithms that attempt deployments prioritising placement of applications to devices that feature with less free resources.

From an alternative viewpoint, [16] gave a Mixed-Integer Non-Linear Programming (MINLP) formulation of the problem of placing application components so to satisfy end-to-end delay constraints. The problem is then solved by linearisation into a Mixed-Integer Linear Programming (MILP), showing potential improvements in latency, energy consumption and costs for routing and storage that the Fog might bring. Also [29] adopted an ILP formulation of the problem of allocating computation to Fog nodes so to optimise time deadlines on application execution. A simple linear model for the Cloud costs is also taken into account. Finally, dynamic programming (e.g., [27]), genetic algorithms (e.g., [29]) and deep learning (e.g., [31]) were exploited promisingly in some recent works.

Overall, to the best of our knowledge, no previous work included a quantitative assessment of the security level of candidate Fog application deployments.

3 Methodology Overview

The OpenFog Consortium [1] highlighted the need for Fog computing platforms to guarantee privacy, anonymity, integrity, trust, attestation, verification and measurement. Whilst security control frameworks exist for Cloud computing scenarios (e.g., the EU Cloud SLA Standardisation Guidelines [2] or the ISO/IEC 19086), to the best of our knowledge, no standard exists yet that defines security objectives for Fog application deployments. Based on recent surveys about security aspects in Fog computing (i.e., [21], [22], [25]), we devised a simple example of taxonomy[1] (Figure 1) of security features that can be offered by Cloud and Fog nodes and therefore used for reasoning on the security levels of given Fog application deployments.

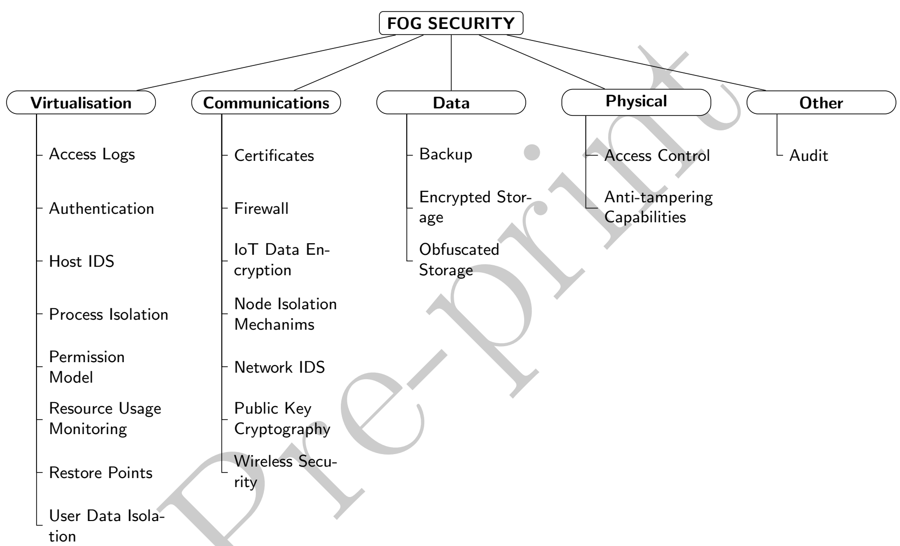

Fig. 1. An example of taxonomy of security capabilities in Fog computing.

Security features that are common with the Cloud might assume renewed importance in Fog scenarios, due to the limited capabilities of the available devices. For instance, guaranteeing physical integrity of and user data isolation at an access point with Fog capabilities might be very difficult. Apropos, the possibility to encrypt or obfuscate data at Fog nodes, along with encrypted IoT communication and physical anti-tampering machinery, will be key to protect those application deployments that need data privacy assurance.

Figure 2 shows the ingredients needed to perform the security assessment by means of the SecFog methodology. On the one hand, we assume that infrastruc-

[1] The proposed taxonomy can be easily modified, extended and refined so as to include new security categories and third-level security features as soon as normative security frameworks will get established for the Fog.

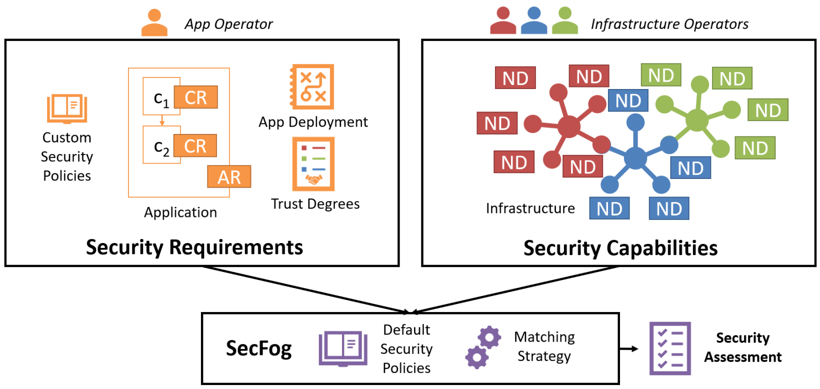

Fig. 2. Bird's-eye view of SecFog.

ture operators declare the *security capabilities* featured by their nodes[2]. Namely, for each node she is managing, the operator publishes a Node Descriptor (ND) featuring a list of the node security capabilities along with a declared measure of their reliability (in the range $[0, 1]$), as shown in Figure 4. On the other hand, based on the same common vocabulary, application operators can define (non-trivial) *custom security policies*. Such properties can complete or override a set of *default security policies* available in SecFog implementation. Custom security policies can be either existing ones, inferred from the presence of certain node capabilities, or they can be autonomously specified/enriched by the application deployers, depending on business-related considerations.

For instance, one can derive that application components deployed to nodes featuring Public Key Cryptography capabilities can communicate through End-to-End Secure channel. A different stakeholder might also require the availability of Certificates at both end-point to consider a channel End-to-End Secure. Similarly, one can decide to infer that a node offering Backup capabilities together with Encrypted Storage or Obfuscated Storage can be considered a Secure Storage provider. Custom and default properties are used, along with ground facts, to specify the *security requirements* of a given application as Component Requirements (CR) and Application Requirements (AR), or both. For instance, application operators can specify that a certain component c is securely deployed to node n when n features Secure Storage and when the communication with component c' happens over an End-to-End Secure channel.

Finally, the security level of an *application deployment* can be assessed by matching the security requirements of the application with the security capabilities featured by the infrastructure and by multiplying the reliability of all exploited security capabilities, weighting them as per *trust degrees*, which may

[2] For the sake of simplicity, in this paper, we assume that operators exploit the vocabulary of the example taxonomy in Figure 1. In reality, different operators can employ different vocabulary and then rely on mediation mechanisms.

be assigned by application deployers to each infrastructure operator. This last step can be used both to assess the security level of a single (possibly partial) input application deployment and to generate and test all eligible deployments according to the declared security requirements. We now go through a motivating example that we will retake later on by exploiting the SecFog prototype.

3.1 Motivating Example

We retake the application example of [6]. Consider a simple Fog application (Figure 3) that manages fire alarm, heating and A/C systems, interior lighting, and security cameras of a smart building. The application consists of three microservices:

- IoTController, interacting with the connected cyber-physical systems,
- DataStorage, storing all sensed information for future use and employing machine learning techniques to update sense-act rules at the IoTController so to optimise heating and lighting management based on previous experience and/or on people behaviour, and
- Dashboard, aggregating and visualising collected data and videos, as well as allowing users to interact with the system.

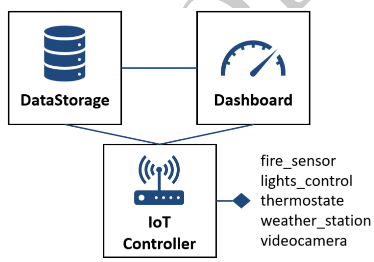

Fig. 3. Fog application.

Each microservice represents an independently deployable component of the application [24] and has hardware and software requirements[3] in order to function properly. Application components must cooperate so that well-defined levels of service are met at runtime. Hence, communication links supporting component-component and component-thing interactions should provide suitable end-to-end latency and bandwidth.
Figure 4 shows the infrastructure – two Cloud data centres, three Fog nodes – to which the smart building application is deployed. For each node, the available security capabilities and their reliability (as declared by the infrastructure operator) are listed in terms of the taxonomy of Figure 1.

[3] For the sake of readability, we omit the application requirements. The interested reader can find all the details in [6].

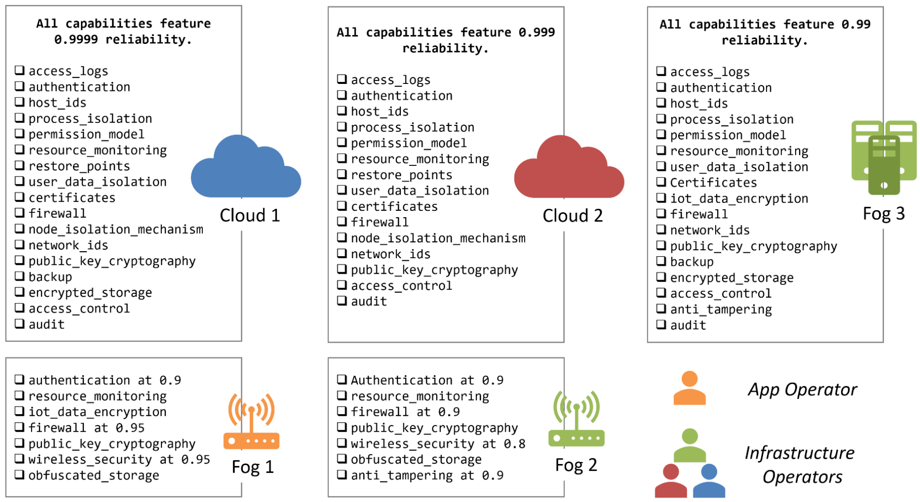

Fig. 4. Fog infrastructure: security view.

Table 1 lists all the deployments of the given application to the considered infrastructure which meet all set software, hardware and network QoS requirements, as they are found by FogTorchΠ in [6]. For each deployment, FogTorchΠ outputs the QoS-assurance (i.e., the likelihood it will meet network QoS requirements), an aggregate measure of Fog resource consumption, and an estimate of the monthly cost for keeping the deployment up and running. Deployments annotated with * are only available when Fog 2 features a 4G connection which costs, however, 20 € a month in addition to the costs reported in Table 1.

In [6], the deployments $\Delta 2$ and $\Delta 16$ are selected as the best candidates depending on the type of mobile connection (i.e., 3G vs 4G) available at Fog 2. As the majority of the existing approaches for application placement, [6] focuses on finding deployments that guarantee application functionality and end-user preferences, currently ignoring security aspects in the featured analysis.

Nevertheless, the application operators are able to define the following Component Requirements:

- IoTController requires Physical Security guarantees (i.e., Access Control $\vee$ Anti-tampering Capabilities) so to avoid that temporarily stored data can be physically stolen from the deployment node,
- DataStorage requires Secure Storage (viz., Backup $\wedge$ (Obfuscated Storage $\vee$ Encrypted Storage)), the availability of Access Logs, a Network IDS in place to prevent distributed Denial of Service (dDoS) attacks, and
- Dashboard requires a Host IDS installed at the deployment node (e.g., an antivirus software) along with a Resource Usage Monitoring to prevent interactions with malicious software and to detect anomalous component behaviour.

Table 1. Eligible deployments of the example application.

Dep. ID	IoTController	DataStorage	Dashboard	QoS	Resources	Cost
$\Delta 1$	Fog 2	Fog 3	Cloud 2	98.6%	48.4%	€856.7
$\Delta 2$	Fog 2	Fog 3	Cloud 1	98.6%	48.4%	€798.7
$\Delta 3$	Fog 3	Fog 3	Cloud 1	100%	48.4%	€829.7
$\Delta 4$	Fog 2	Fog 3	Fog 1	100%	59.2%	€844.7
$\Delta 5$	Fog 1	Fog 3	Cloud 1	96%	48.4%	€837.7
$\Delta 6$	Fog 3	Fog 3	Cloud 2	100%	48.4%	€887.7
$\Delta 7$	Fog 3	Fog 3	Fog 2	100%	59.2%	€801.7
$\Delta 8$	Fog 3	Fog 3	Fog 1	100%	59.2%	€875.7
$\Delta 9$	Fog 1	Fog 3	Cloud 2	96%	48.4%	€895.7
$\Delta 10$	Fog 1	Fog 3	Fog 2	100%	59.2%	€809.7
$\Delta 11$	Fog 1	Fog 3	Fog 1	100%	59.2%	€883.7
$\Delta 12^*$	Fog 2	Cloud 2	Fog 1	94.7%	16.1%	€870.7
$\Delta 13^*$	Fog 2	Cloud 2	Cloud 1	97.2%	5.4%	€824.7
$\Delta 14^*$	Fog 2	Cloud 2	Cloud 2	98.6%	5.4%	€882.7
$\Delta 15^*$	Fog 2	Cloud 1	Cloud 2	97.2%	5.4%	€785.7
$\Delta 16^*$	Fog 2	Cloud 1	Cloud 1	98.6%	5.4%	€727.7
$\Delta 17^*$	Fog 2	Cloud 1	Fog 1	94.7%	16.1%	€773.7

Furthermore, the Application Requirements require guaranteed end-to-end encryption among all components (viz., all deployment nodes should feature Public Key Cryptography) and that deployment nodes should feature an Authentication mechanism. Finally, application operators assign a trust degree of 80% to the infrastructure providers of Cloud 1 and Cloud 2, and of 90% to the infrastructure providers of Fog 3 and Fog 2. Naturally, they consider their management of Fog 1 completely trustable.

4 Proof-of-Concept

Being SecFog a declarative methodology based on probabilistic reasoning about declared infrastructure capabilities and security requirements, it was natural to prototype it relying on probabilistic logic programming. To implement both the model and the matching strategy we used a language called *ProbLog* [10]. ProbLog is a Python package that permits writing logic programs that encode complex interactions between large sets of heterogeneous components, capturing the inherent uncertainties that are present in real-life situations. Problog programs are composed of *facts* and *rules*. The facts, such as

```
p::f.
```

represent a statement `f` which is true with probability `p`[4]. The rules, like

```
r :- c1, ... , cn.
```

[4] A fact declared simply as `f.` is assumed to be true with probability 1.

represent a property `r` inferred when $\texttt{c1} \wedge \cdots \wedge \texttt{cn}$ hold[5]. ProbLog programs are logic programs in which some of the facts are annotated with (their) probabilities. Each program defines a probability distribution over logic programs where a fact `p::f.` is considered true with probability `p` and false with probability $1 - \texttt{p}$. The ProbLog engine [11] determines the success probability of a query `q` as the probability that `q` has *a* proof, given the distribution over logic programs.

Our prototype offers three main default security policies that can be used to compose more complex application security requirements. First

```
secure(C, N, D) :-
    member(d(C,N), D),
    node(N, Op),
    trustable(Op).
```

that checks if a component `C` is actually deployed to an existing node `N` (as per deployment `D`) and that the infrastructure operator `Op` managing `N` is trustable according to the application operator. Then

```
secureApp(A,D) :-
    app(A,L),
    deployment(L,D),
    secureComponents(A,L,D).

secureComponents(A, [], _).
secureComponents(A, [C|Cs],D) :-
    secureComponent(C,N,D),
    secureComponents(A,Cs,D).
```

that checks whether, according to an input deployment `D`, each component of a given application `A` can be securely deployed, i.e. if `secureComponent(C, N, D)` holds for all components `C` of `A`. The application operator is therefore asked to define a `secureComponent(C, N, D)` for each of the application components, always including the default predicate `secure(C, N, D)`.

4.1 Motivating Example Continued

In this section, we retake the example of Section 3.1 and we show how ProbLog permits to naturally express both security capabilities of an infrastructure and security requirements of an application.

Node Descriptors can be expressed by listing ground facts, possibly featuring a probability that represents their reliability according to the infrastructure provider. For instance, `fog1` directly operated by the application operator `appOp` is described as

```
node(fog1,appOp).
0.9::authentication(fog1).
```

[5] Both `r` and {`ci`} can have variable (upper-case) or constant (lower-case) input parameters.

```
resource_monitoring(fog1).
iot_data_encryption(fog1).
0.95::firewall(fog1).
public_key_cryptography(fog1).
0.95::wireless_security(fog1).
obfuscated_storage(fog1).
```

All the Node Descriptors made following this template form a description of the *security capabilities* available in the infrastructure.
Application operators can define the topology of an application by specifying an identifier and the set of its components. For instance, the application of Figure 3 can be simply denoted by the fact

```
app(smartbuilding, [iot_controller, data_storage, dashboard]).
```

Then, they can define the *security requirements* of the application both as Component Requirements and Application Requirements. In our example, the Component Requirements can be simply declared as

```
secureComponent(iot_controller, N, D) :-
    physical_security(N),
    secure(iot_controller, N, D).

secureComponent(data_storage, N, D) :-
    secure_storage(N),
    access_logs(N),
    network_ids(N),
    secure(data_storage, N, D).

secureComponent(dashboard, N, D) :-
    host_ids(N),
    resource_monitoring(N),
    secure(dashboard, N,D).
```

where the custom security policies `physical_security(N)` and `secure_storage(N)` are defined as

```
secure_storage(N) :-
    backup(N),
    (encrypted_storage(N); obfuscated_storage(N)).

physical_security(N) :- anti_tampering(N); access_control(N).
```

Analogously, the Application Requirements that concern the application as a whole can be specified by extending the default policy `secureApp(A,D)` as follows

```
mySecureApp(A,D) :-
    secureApp(A,D),
    deployment(L,D),
    extras(D).
```

where the custom security policy `extras(N)` checking for Public Key Cryptography and Authentication at all nodes are (recursively) defined as

```
extras([]).
extras([d(C,N)|Ds]) :-
    public_key_cryptography(N),
    authentication(N),
    extras(Ds).
```

Finally, application operators can express their *trust degrees* towards each infrastructure operator as the probability of trusting it (i.e., $t \in [0, 1]$). In our example, we have

```
0.8::trustable(cloudOp1).
0.8::trustable(cloudOp2).
0.9::trustable(fogOp).
trustable(appOp).
```

Our prototype can be used to find (via a *generate* & *test* approach) all deployments that satisfy the security requirements of the example application to a given infrastructure, by simply issuing the query[6]

```
query(mySecureApp(smartbuilding,L)).
```

As shown in Figure 5, relying on ProbLog out-of-the-box algorithms, SecFog prototype returns answers to the query along with a value in $[0, 1]$ that represents the aggregate *security level* of the inferred facts, i.e. the probability that a deployment can be considered secure both according to the declared reliability of the infrastructure capabilities and to the trust degree of the application operator in each exploited infrastructure provider.

If the application operator is only considering security as a parameter to lead her search, she would try to maximise the obtained metric and, most probably, deploy all three components to Fog 3. However, security might need to be considered together with other parameters so to find a trade-off among them. In the next section, we propose a simple multi-objective optimisation and we apply it to our motivating example.

5 Multi-Objective Optimisation

Naturally, the quantitative results obtained with ProbLog can be used to optimise the security level of any application deployment, by simply taking the maximum value for our query. As we will show over an example in the next section, it is possible to exploit the SecFog methodology to optimise the security level together with other metrics. In this work, as in [14], given a deployment Δ, we will try to optimise the objective function

$$r(\Delta) = \sum_{m \in M} \omega_m \cdot \widehat{m(\Delta)}$$

[6] Naturally, it is also possible to specify one particular deployment and assess its security level only.

```
mySecureApp(smartbuilding,[d(iot_controller,cloud1), d(data_storage,cloud1), d(dashboard,cloud1)]): 0.79928029
mySecureApp(smartbuilding,[d(iot_controller,cloud1), d(data_storage,cloud1), d(dashboard,cloud2)]): 0.63699776
mySecureApp(smartbuilding,[d(iot_controller,cloud1), d(data_storage,cloud1), d(dashboard,fog3)]): 0.69114513
mySecureApp(smartbuilding,[d(iot_controller,cloud1), d(data_storage,fog3), d(dashboard,cloud1)]): 0.67752684
mySecureApp(smartbuilding,[d(iot_controller,cloud1), d(data_storage,fog3), d(dashboard,cloud2)]): 0.53996463
mySecureApp(smartbuilding,[d(iot_controller,cloud1), d(data_storage,fog3), d(dashboard,fog3)]): 0.66417689
mySecureApp(smartbuilding,[d(iot_controller,cloud2), d(data_storage,cloud1), d(dashboard,cloud1)]): 0.63757163
mySecureApp(smartbuilding,[d(iot_controller,cloud2), d(data_storage,cloud1), d(dashboard,cloud2)]): 0.63642441
mySecureApp(smartbuilding,[d(iot_controller,cloud2), d(data_storage,cloud1), d(dashboard,fog3)]): 0.55131415
mySecureApp(smartbuilding,[d(iot_controller,cloud2), d(data_storage,fog3), d(dashboard,cloud1)]): 0.54045108
mySecureApp(smartbuilding,[d(iot_controller,cloud2), d(data_storage,fog3), d(dashboard,cloud2)]): 0.67448315
mySecureApp(smartbuilding,[d(iot_controller,cloud2), d(data_storage,fog3), d(dashboard,fog3)]): 0.66238504
mySecureApp(smartbuilding,[d(iot_controller,fog2), d(data_storage,cloud1), d(dashboard,cloud1)]): 0.5827336
mySecureApp(smartbuilding,[d(iot_controller,fog2), d(data_storage,cloud1), d(dashboard,cloud2)]): 0.46441781
mySecureApp(smartbuilding,[d(iot_controller,fog2), d(data_storage,cloud1), d(dashboard,fog3)]): 0.55988355
mySecureApp(smartbuilding,[d(iot_controller,fog2), d(data_storage,fog3), d(dashboard,cloud1)]): 0.54885163
mySecureApp(smartbuilding,[d(iot_controller,fog2), d(data_storage,fog3), d(dashboard,cloud2)]): 0.54687823
mySecureApp(smartbuilding,[d(iot_controller,fog2), d(data_storage,fog3), d(dashboard,fog3)]): 0.67268088
mySecureApp(smartbuilding,[d(iot_controller,fog3), d(data_storage,cloud1), d(dashboard,cloud1)]): 0.70503715
mySecureApp(smartbuilding,[d(iot_controller,fog3), d(data_storage,cloud1), d(dashboard,cloud2)]): 0.56188935
mySecureApp(smartbuilding,[d(iot_controller,fog3), d(data_storage,cloud1), d(dashboard,fog3)]): 0.69114513
mySecureApp(smartbuilding,[d(iot_controller,fog3), d(data_storage,fog3), d(dashboard,cloud1)]): 0.67752684
mySecureApp(smartbuilding,[d(iot_controller,fog3), d(data_storage,fog3), d(dashboard,cloud2)]): 0.67509079
mySecureApp(smartbuilding,[d(iot_controller,fog3), d(data_storage,fog3), d(dashboard,fog3)]): 0.83038718
```

Fig. 5. Results of the motivating example.

where M is the set of metrics to be optimised, ω_m is the weight[7] assigned to each metrics (so that $\sum_{m \in M} \omega_m = 1$) and $\widehat{m(\Delta)}$ is the normalised value of metric m for deployment Δ, which – given the set D of candidate deployments – is computed as:

- $\widehat{m(\Delta)} = \frac{m(\Delta) - \min_{d \in D}\{m(d)\}}{\max_{d \in D}\{m(d)\} - \min_{d \in D}\{m(d)\}}$ when the $m(\Delta)$ is to be maximised, and
- $\widehat{m(\Delta)} = \frac{\max_{d \in D}\{m(d)\} - m(\Delta)}{\max_{d \in D}\{m(d)\} - \min_{d \in D}\{m(d)\}}$ when $m(\Delta)$ is to be minimised.

Therefore, since we assumed that the higher the value of $r(\Delta)$ the better is deployment Δ, we will choose $\overline{\Delta}$ such that $r(\overline{\Delta}) = \max_{\Delta \in D}\{r(\Delta)\}$. In what follows, we solve the motivating example by employing this optimisation technique on all attributes of Table 1 along with the security levels computed in Section 4.

5.1 Motivating Example Continued

In our motivating example, we will attempt to maximise QoS-assurance and security, whilst minimising cost (in which we include the cost for the 4G connection at Fog 2 when needed). However, different application operators may want to either maximise or minimise the Fog resource consumption of their deployment, i.e. they may look for a Fog-ward or for a Cloud-ward deployment. Hence, concerning this parameter, we will consider both situations. Table 2 show the values of the Fog-ward (i.e., $r_F(\Delta)$) and of the Cloud-ward (i.e., $r_C(\Delta)$) objective function.

[7] For the sake of simplicity, we assume here $\omega_m = \frac{1}{|M|}$, which can be tuned differently depending on the needs of the application operator.

Table 2. Ranking of eligible deployments.

Dep. ID	IoTController	DataStorage	Dashboard	$r_F(\Delta)$	$r_C(\Delta)$
$\Delta 1$	Fog 2	Fog 3	Cloud 2	0.53	0.28
$\Delta 2$	Fog 2	Fog 3	Cloud 1	0.63	0.38
$\Delta 3$	Fog 3	Fog 3	Cloud 1	0.85	0.60
$\Delta 6$	Fog 3	Fog 3	Cloud 2	0.75	0.50
$\Delta 15^*$	Fog 2	Cloud 1	Cloud 2	0.15	0.40
$\Delta 16^*$	Fog 2	Cloud 1	Cloud 1	0.51	0.76

In the Fog-ward case, when looking for the best trade-off among QoS-assurance, resource consumption, cost and security level, the most promising deployment is not $\Delta 2$ anymore (as it was in [5]). Indeed, $\Delta 3$ scores a much better ranking when compared to $\Delta 2$. Furthermore, in the Fog-ward case, the 4G upgrade at Fog 2, which makes it possible to enact $\Delta 15$ and $\Delta 16$, is not worth the investment due to the low score of both deployments. Conversely, in the Cloud-ward case (even though $\Delta 3$ would still be preferable), $\Delta 16$ features a good ranking value, despite requiring to upgrade the connection available at Fog 2.

6 Concluding Remarks

In this paper, we proposed a declarative methodology, SecFog, which can be used to assess the security level of multi-component application deployments to Fog computing infrastructures. With a proof-of-concept implementation in ProbLog, we have shown how SecFog helps application operators in determining secure deployments based on specific application requirements, available infrastructure capabilities, and trust degrees in different Fog and Cloud providers. We have also shown how SecFog can be used synergically with other predictive methodologies to perform multi-objective optimisation of security along with other metrics (e.g., deployment cost, QoS-assurance, resource usage). In our future work we plan to:

- enhance SecFog by combining it with existing strategies that have been used to quantify trust degrees (e.g., Bayesian or Dempster–Shafer theories as in [34]) based on direct experience, possibly considering also the mobility of Fog nodes and IoT devices,
- evaluate the possibility to use SecFog with meta-heuristic optimisation techniques (e.g., genetic or swarm intelligence algorithms), also taming the time complexity of the generate & test approach we prototyped, and
- further engineer our proof-of-concept implementation and show its applicability to actual use cases (e.g., based on the Fog application of [7]).

References

1. OpenFog Consortium. http://www.openfogconsortium.org/

2. EU Cloud SLA Standardisation Guidelines (2014), https://ec.europa.eu/digital-single-market/en/news/cloud-service-level-agreement-standardisation-guidelines
3. Belle, V.: Logic meets probability: towards explainable ai systems for uncertain worlds. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI. pp. 19–25 (2017)
4. Brogi, A., Forti, S.: QoS-Aware Deployment of IoT Applications Through the Fog. IEEE Internet of Things Journal **4**(5), 1185–1192 (Oct 2017)
5. Brogi, A., Forti, S., Ibrahim, A.: How to best deploy your Fog applications, probably. In: Rana, O., Buyya, R., Anjum, A. (eds.) Proceedings of 1st IEEE Int. Conference on Fog and Edge Computing (2017)
6. Brogi, A., Forti, S., Ibrahim, A.: Deploying fog applications: How much does it cost, by the way? In: Proceedings of the 8th International Conference on Cloud Computing and Services Science. pp. 68–77. SciTePress (2018)
7. Brogi, A., Forti, S., Ibrahim, A., Rinaldi, L.: Bonsai in the fog: An active learning lab with fog computing. In: Fog and Mobile Edge Computing (FMEC), 2018 Third International Conference on. pp. 79–86. IEEE (2018)
8. Choo, K.K.R., Lu, R., Chen, L., Yi, X.: A foggy research future: Advances and future opportunities in fog computing research (2018)
9. Dastjerdi, A.V., Buyya, R.: Fog computing: Helping the internet of things realize its potential. Computer **49**(8), 112–116 (Aug 2016)
10. De Raedt, L., Kimmig, A.: Probabilistic (logic) programming concepts. Machine Learning **100**(1), 5–47 (2015)
11. De Raedt, L., Kimmig, A., Toivonen, H.: Problog: A probabilistic prolog and its application in link discovery. In: Proceedings of the 20th International Joint Conference on Artifical Intelligence. pp. 2468–2473 (2007)
12. Goettelmann, E., Dahman, K., Gateau, B., Dubois, E., Godart, C.: A security risk assessment model for business process deployment in the cloud. In: Services Computing (SCC), 2014 IEEE International Conference on. pp. 307–314. IEEE (2014)
13. Guerrero, C., Lera, I., Juiz, C.: A lightweight decentralized service placement policy for performance optimization in fog computing. Journal of Ambient Intelligence and Humanized Computing (Jun 2018)
14. Guerrero, C., Lera, I., Juiz, C.: Resource optimization of container orchestration: a case study in multi-cloud microservices-based applications. The Journal of Supercomputing **74**(7), 2956–2983 (Jul 2018)
15. Gupta, H., Vahid Dastjerdi, A., Ghosh, S.K., Buyya, R.: iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. Software: Practice and Experience **47**(9), 1275–1296 (2017)
16. Hamid Reza Arkian, Abolfazl Diyanat, A.P.: Mist: Fog-based data analytics scheme with cost-efficient resource provisioning for IoT crowdsensing applications. Journal of Network and Computer Applications **82**, 152 – 165 (2017)
17. Hong, H.J., Tsai, P.H., Hsu, C.H.: Dynamic module deployment in a fog computing platform. In: 2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS). pp. 1–6 (Oct 2016)
18. Kaur, A., Singh, M., Singh, P., et al.: A taxonomy, survey on placement of virtual machines in cloud. In: 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS). pp. 2054–2058. IEEE (2017)
19. Luna, J., Taha, A., Trapero, R., Suri, N.: Quantitative reasoning about cloud security using service level agreements. IEEE Transactions on Cloud Computing **5**(3), 457–471 (July 2017)

20. Mahmud, R., Ramamohanarao, K., Buyya, R.: Latency-aware application module management for fog computing environments. ACM Transactions on Internet Technology (TOIT) (2018)
21. Mezni, H., Sellami, M., Kouki, J.: Security-aware SaaS placement using swarm intelligence. Journal of Software: Evolution and Process (2018)
22. Mukherjee, M., Matam, R., Shu, L., Maglaras, L., Ferrag, M.A., Choudhury, N., Kumar, V.: Security and privacy in fog computing: Challenges. IEEE Access **5**, 19293–19304 (2017)
23. Nacer, A.A., Goettelmann, E., Youcef, S., Tari, A., Godart, C.: Obfuscating a business process by splitting its logic with fake fragments for securing a multi-cloud deployment. In: Services (SERVICES), 2016 IEEE World Congress on. pp. 18–25. IEEE (2016)
24. Newman, S.: Building microservices: designing fine-grained systems. " O'Reilly Media, Inc." (2015)
25. Ni, J., Zhang, K., Lin, X., Shen, X.: Securing fog computing for internet of things applications: Challenges and solutions. IEEE Comm. Surveys & Tutorials (2017)
26. OpenFog: OpenFog Reference Architecture (2016)
27. Rahbari, D., Nickray, M.: Scheduling of fog networks with optimized knapsack by symbiotic organisms search. In: 2017 21st Conference of Open Innovations Association (FRUCT). pp. 278–283 (Nov 2017)
28. Schoenen, S., Mann, Z.Á., Metzger, A.: Using risk patterns to identify violations of data protection policies in cloud systems. In: International Conference on Service-Oriented Computing. pp. 296–307. Springer (2017)
29. Skarlat, O., Nardelli, M., Schulte, S., Dustdar, S.: Towards qos-aware fog service placement. In: 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC). pp. 89–96 (May 2017)
30. Taneja, M., Davy, A.: Resource aware placement of iot application modules in fog-cloud computing paradigm. In: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). pp. 1222–1228 (May 2017)
31. Tang, Z., Zhou, X., Zhang, F., Jia, W., Zhao, W.: Migration modeling and learning algorithms for containers in fog computing. IEEE Transactions on Services Computing (2018)
32. Varshney, P., Simmhan, Y.: Demystifying Fog Computing: Characterizing Architectures, Applications and Abstractions. In: 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC). pp. 115–124 (2017)
33. Wang, S., Zafer, M., Leung, K.K.: Online placement of multi-component applications in edge computing environments. IEEE Access **5**, 2514–2533 (2017)
34. Wei, Z., Tang, H., Yu, F.R., Wang, M., Mason, P.: Security enhancements for mobile ad hoc networks with trust management using uncertain reasoning. IEEE Transactions on Vehicular Technology **63**(9), 4647–4658 (Nov 2014)
35. Wen, Z., Yang, R., Garraghan, P., Lin, T., Xu, J., Rovatsos, M.: Fog Orchestration for Internet of Things Services. IEEE Internet Computing **21**(2), 16–24 (2017)
36. Wen, Z., Cała, J., Watson, P., Romanovsky, A.: Cost effective, reliable and secure workflow deployment over federated clouds. IEEE Transactions on Services Computing **10**(6), 929–941 (2017)
37. Zhang, P., Zhou, M., Fortino, G.: Security and trust issues in fog computing: A survey. Future Generation Computer Systems **88**, 16–27 (2018)