FORTIFIER: a FORmal disTrIbuted Framework to Improve the dEtection of thReAtening objects in baggage

Pablo C. Cañizares, Mercedes G. Merayo and Alberto Núñez

Departamento de Sistemas Informáticos y Computación, Universidad Complutense de Madrid, Madrid, Spain

ABSTRACT
Currently, security breaches in public places like airports and official buildings are a major concern by both governmental and corporative organizations. In these situations, X-ray devices must scan a vast amount of baggage in a short time frame. Hence, deploying scanners that automatize the task of detecting suspicious artefacts becomes of vital importance to prevent from threats. In this paper we present FORTIFIER, a formal distributed framework designed to detect suspicious artefacts. This approach consists in the integration of several image detection algorithms executed in a distributed environment, which are aimed to detect a wide spectrum of weapons like guns, knives and bombs. The main core of our proposed framework for recognizing suspicious artefacts is divided in different phases, where each one is modelled with a specific finite state machine (FSM). Several FSMs are combined to detect different artefacts. We also present a case of study where some performance experiments are carried out for analysing the scalability of FORTIFIER. Initially, FORTIFIER is deployed in a single cloud environment. Once the main features that have a significant impact on the overall system performance are analysed, our proposed framework is deployed in a multi-cloud environment.

ARTICLE HISTORY
Received 5 May 2017
Accepted 25 May 2017

KEYWORDS
Formal modelling; image recognition; cloud systems

1. Introduction

The manual detection of suspicious artefacts carried out by human operators is a complex and arduous task. The luggage usually consists of a large quantity of items, usually overlapped, which hampers the detection of this kind of malicious objects. Also, the quantity of baggage that contain threatening objects represents a very low percentage of the total, and the technological support provided to the operators is very reduced. Therefore, at peak times, the worker must instantly decide whether or not a luggage can be considered as suspicious. Since each operator must analyse several baggages, the human error level can be very high even if they have been intensively trained.

After the 11/9 tragedy, security has become a national priority for governments around the world. Consequently, high security measures have been imposed on both governmental and corporative facilities to ensure the integrity of citizens. It is worth emphasizing the
case of the airports, where only the U.S. government generated an annual $700 million market (107-71 2001) in the deployment of Explosives Detection Systems (EDS) (Murray & Riordan, 1995; Singh & Singh, 2003), which are based on X-ray imaging for scanning baggage (see Figure 1).

For all of these reasons, the massive deployment of EDS has arisen the interest of the scientific community. Several automatic detection techniques have been reported in the literature (Mery, 2015a; Wells & Bradley, 2012), such as artificial neural networks (Liu & Wang, 2008; Singh & Singh, 2004), support vector machines (SVM) (Franzel, Schmidt, & Roth, 2012; Nercessian, Panetta, & Agaian, 2008) and novel multiple-view approaches (Mery, 2015b; Mery, Riffo, Zuccar, & Pieringer, 2017; Uroukov & Speller, 2015). All these techniques, especially the multiple-view ones (see Figure 2), have high detection rates for processing all type of threatening artefacts such as guns, weapons, bombs and knives, which represents a beneficial contribution to protect the human life. However, these techniques are designed to be executed in a single machine. Unfortunately, this solution provides a low performance, lacks of fault tolerance and it is overexposed to risks such as computer attacks.

During the last years, cloud computing systems are increasing their role due to the wide adoption of new computer networks and the fast evolution of computing technologies. Cloud computing can be defined as a paradigm that provides access to a flexible and on-demand computing infrastructure, by allowing the user to deploy virtual machines for a specific time slot giving the illusion of unlimited resources. A very clear proof of this fact is that very important companies like Amazon, Google, Dell, IBM, and Microsoft are investing billions of dollars in order to provide their own cloud solutions.

Currently, several factors motivate the interest of migrating and deploying systems in the cloud, like the possibility of accessing to a flexible computing system with the possibility of varying the number of CPUs and the memory size in runtime. The leap from private data-centres and local clusters to this new paradigm has been imposed in many research areas, it being the main reason for the evolution of computational needs. Also, the lack of infrastructure and administration duties results in ease off management and

![Figure 1](image-url). X-ray image for detecting guns in baggage (Mery, 2015b).
overall cost reductions, where users are released from those tasks for managing physical servers or storage devices.

In this paper we introduce a distributed framework, called FORTIFIER, to detect suspicious artefacts. In order to avoid security risks, it is important to incorporate mechanisms to increase the confidence on the correctness of the system. It is widely recognized that the combination of formal methods and testing techniques is very beneficial (Cavalli, Higashino, & Núñez, 2015; Gaudel, 1995; Hierons et al., 2009; Hierons, Bowen, & Harman, 2008; Hierons, Merayo, & Núñez, 2016, 2017; Veanes et al., 2008) and industry is becoming aware of the importance of using formal approaches (Grieskamp, Kicillof, Stobie, & Braberman, 2011). The system proposed in this work present interconnected components and requires a framework that allow to analyse the correctness of the communication among them. We have adopted a formal approach that allows to model our system using a formalism based on communicating finite state machines (Merayo & Núñez, 2015). Each component of FORTIFIER is given by a finite state machine. In order to ensure the correct behaviour of the proposal, a specific set of properties that involve the communication among the components has been designed and checked against our model. A complete case study has been carried out. The system has been implemented and deployed along several cloud systems in a simulated environment conducted by using the SIMCAN simulation platform (Núñez, Fernández, Filgueira, García, & Carretero, 2012). It has allowed us to analyse both the correctness of the implementation and the performance scalability of FORTIFIER.

This paper extends and enhances our previous work (Cañizares, Merayo, & Núñez, 2016). Specifically, we can mention the following contributions.

- We have included an extensive review of the main proposals to detect suspicious artefacts.
- We have designed a new set of communicating invariants that represent different behaviours that the system must fulfill.
- We have extended the evaluation process with new experiments. We report a detailed analysis of the overall system performance obtained by the application of our proposal, based on the results.

The rest of the paper is structured as follows. Section 2 reviews the related work. Section 3 presents the formal framework used in this paper. Next, in Section 4 we describe the
proposed distributed scheme. Section 5 presents experimental results. Finally, in Section 6 we present the conclusions and some lines of future work.

2. Related work

During the last years, several contributions focusing the detection of suspicious artefacts, by analysing X-ray images of luggage, can be found in the literature. In general, these artefacts could be threatening for the humans health, which is a general concern. The existing contributions can be categorized in two main groups: single-view (Liu & Wang, 2008; Singh & Singh, 2004) and multiple-view (Mery, 2015b; Uroukov & Speller, 2015).

The single-view techniques are those whose main goal is to detect threatening objects by analysing a single image. In this field, there exists a large quantity of contributions (Liu & Wang, 2007; Paranjape, Sluser, & Runtz, 1998; Turcsany, Mouton, & Breckon, 2013). Among them, let us remark two main subsets, those based on artificial neural network and those based on SVM. In the field of the artificial neural networks, Singh & Singh (2004) presented a methodology for optimizing image segmentation algorithms in an automatic way. The proposed methodology uses several image properties, such as intra-cluster distances, colour purity and gradient strength to train an artificial neural network that is used to predict the acceptance degree of the proposed solution. Finally, the authors performs an empirical study that shows the suitability of the proposal. At the same research line, Liu & Wang (2008) proposed a classification system based on artificial neural networks and fuzzy logic to detect explosives in X-ray images. Then, a multi-level fuzzy classifier and a parallel artificial neural network are used to improve the accuracy level of the proposed system. In the field of SVMs, Nercessian et al. (2008) presented an automatic system for the detection of threatening objects in X-ray luggage images. The system uses segmentation and feature vectors, which are considered as the pillars of the artificial intelligent system. An experimental study has been include to analyse and detect handguns, which shows both the effectiveness of the system for detect this kind of suspicious objects and the suitability of the algorithm for real-time applications. Al-Qubaa & Tian (2012) presented a weapon detection system based on time and frequency extraction techniques. The main idea of the proposal is that each weapon has a unique electromagnetic fingerprint, determined by its size, shape and physical composition. The empirical study carried out in this paper shows the potential and efficiency of the system by detecting guns and non-gun objects in controlled and non-controlled environments.

In the last 5 years there have been several approaches based on a novel technique known as multiple-view (Baştan, 2015; Mery, 2013, 2014). The multiple-view techniques are those whose main goal is to detect threatening objects analysing a transition of multiple views. Franzel et al. (2012) presented an automatic object detection approach for multi-view X-ray image data. This proposal is two folded, on the one hand the system analyses the variations of the X-ray images obtained from external analysers and adapts existing appearance-based object detection approaches to the X-ray image data. In this way, the authors intends to decrease distortions and to increase the feature set. On the other hand, the system uses a multi camera detection approach to analyse single-view images and multiple-view images, which improves the effectiveness of the proposed systems using the mutual reinforcement of geometrically consistent hypotheses. The experimental phase evaluates the proposed method detecting handguns in carry-on
luggage. Mery presented a multiple-view methodology to identify and extract features of a complex artefact (2015b). The methodology is based on five steps, image acquisition, geometric model estimation, single-view detection, multiple-view detection and analysis. The experimental study that has been carried out to validate the methodology shows that this proposal outperforms the existing representative approaches existing in the state of the art. Uroukov & Speller (2015) proposed a system to detect suspicions objects during the scanning process using textural signatures to recognize a wide spectrum of materials. In this work, the authors carry out an experimental study where several images of industrial standards have been filtered using a directional Gabor-type approach and analysed using a diverse spectrum of range and orientation. In the experimental phase it was found that different materials could be characterized in terms of the frequency range and orientation of the filters. More recently, Mery et al. (2017) presented an automated multiple-view method to recognize objects with highly defined shapes and sizes. The proposed method is two folded: the first step is to analyse each view of the sequence, and the next step consist in perform the analysis using the multiple-view image. With the main objective of illustrating the suitability of the proposed method, an experimental study have been carried out in order to recognize regular objects such as clips, springs and razor blades, the results have shown a high level of accuracy.

Although the existing techniques reach a suitable level of accuracy by detecting suspicious artefacts in X-ray images, for the best of our knowledge there not exist a distributed schema based on formal methods and simulation techniques to define and analyse the intended system. For this reason, we consider that the contributions proposed in this paper are suitable for the initial stages of the distributed detection systems development, decreasing the quantity of errors due to formal nature of the framework and the post-analysis performed with the simulation tool.

3. Formal framework

In this section we review the framework used for specifying and testing complex systems (Merayo & Núñez, 2015) that has been used to model and specify FORTIFIER. In addition, we introduce some extensions to the finite state machine model, that allows us to define and check the correctness of communications between components of the system.

3.1. Finite state machines

Finite state machines, in short FSM, are one of the formalism widely used to formally specify systems. We have chosen them to specify our system because they are well known and their definition and semantics are very simple.

Definition 3.1  An FSM is a tuple $M = (S, s_{in}, I, O, T_r)$ where $S$ is a finite set of states, $s_{in}$ is the initial state of the machine, $I$ is the set of input actions, $O$ is the set of output actions, with $I \cap O = \emptyset$, and $T_r$ is the set of transitions, with $T_r \subseteq S \times S \times I \times O$. A transition belonging to $T_r$ is a tuple $(s, s', i, o)$ where $s, s' \in S$ are the initial and final states of the transition, respectively, $i \in I$ is the input action and $o \in O$ is the output action.

We say that $M$ is deterministic if for all state $s$ and input $i$ there exists at most one state $s'$ and one output $o'$ such that $(s, s', i, o) \in T_r$. We say that $M$ is input-enabled if for all state $s$ and input $i$ there exists at least one state $s'$ and one output $o'$ such that $(s, s', i, o) \in T_r$. 

---

P. C. CAÑIZARES ET AL.
In the following definition we introduce the concept of *trace*. A trace is a sequence of inputs and outputs pairs that captures the behaviour of a system.

**Definition 3.2** Let $M = (S, s_0, I, O, T_r)$ be a FSM. We say that $(i_1/o_1, \ldots, i_n/o_n)$ is a *trace* of $M$ if there exist $n$ states $s_1, \ldots, s_n \in S$ such that

$$(s_0, s_1, i_1, o_1), (s_1, s_2, i_2, o_2), \ldots, (s_{n-1}, s_n, i_n, o_n) \in T_r$$

with $s_0 = s_0$. We denote by $\langle \rangle$ the empty trace and by $\text{trace}(M)$ the set of all traces of $M$.

**Example 3.3** Let us consider the FSM depicted in Figure 3 presenting a reduced version of the pre-processing image stage. It enhances the visual appearance and improves the manipulation of the image for later stages. The nodes represent the most relevant states of the algorithm, while the arcs represent the relevant transitions performed during the process. The initial state of the machine is $s_1$, corresponding to the point where the image to be pre-processed is received.

Let us consider the transition $(s_1, s_2, \text{ImageRaw}, \text{PreProcI})$. Intuitively, if the machine is the initial state $s_1$ and it receives an input $\text{ImageRaw}$, then it produces the output $\text{PreProcI}$ and the machine changes to state $s_2$. Also, we can observe that $(\text{ImageRaw}/\text{PreProcI}, \text{PPImg}/\text{CheckI}, \text{ContP}/\text{DetectImg})$ is a trace of the system.

Next, we describe the steps required to perform the image pre-processing phase. At the initial state, the system receives an image $\text{ImageRaw}$ and the process $\text{PreProcI}$ is invoked. Once all the pre-processing operations have been performed, the system checks the correctness of the generated image calling the $\text{CheckI}$ process. Finally, the checking process returns a result which shows the diagnostic of the pre-processed image. If the checking process detects that the image has some faults, then the process will be interrupted.

### 3.2. Communicating finite state machines

In order to alleviate hard computational challenges, there is a whole new generation of systems. These systems are usually distributed along the nodes of a network. Thus, the communication between the components of this network becomes a critical factor for
the overall system performance. Unfortunately, the behaviour of these systems cannot be represented by using classical finite state machines and, therefore, it is required to develop new methodologies that allow us both to represent properties related to communications and to establish its correctness.

**Definition 3.4** A Communicating Finite State Machine, in short CFSM, is an FSM with a set of communication channels. A Net Communicating Finite State Machines, in short NETCOM, is a pair $\mathcal{N} = (\mathcal{M}, C)$, where $\mathcal{M} = \{M_1, \ldots, M_n\}$ is a set of CFSMs such that for all $1 \leq i \leq n$ we have that $M_i = (S_i, s_i^0, I_i, O_i, T_i)$ and $C = \{C^a_i : i \leq n \land a \in I_i\}$ represents the set of communication channels, where $C^a_i$ means that $M_i$ can receive the message $a$. We assume that $I_1, \ldots, I_n$ are pairwise disjoint and for all $1 \leq i \leq n$ we have that $I_i \cap O_i = \emptyset$.

Given $I_N = \bigcup_{i=1}^{n} I_i$ and $O_N = \bigcup_{i=1}^{n} O_i$, we define the sets $\text{Shared}_N = I_N \cap O_N$, $\text{envInput}_N = I_N \setminus \text{Shared}_N$ and $\text{envOutput}_N = O_N \setminus \text{Shared}_N$.

A CFSM in a NETCOM can interact both with the environment and with another CFSM, by sending inputs and receiving output actions. Thus, two classes of transitions can be distinguished. On the one hand, external transitions are those labelled with input actions that are received from the environment. On the other hand, internal transitions are those that are triggered by an output produced by the execution of a transition in another CFSM.

The set $\text{shared}_N$ contains those actions allowing the communication between two machines in a net. Those actions belong simultaneously to the set of input actions of a CFSM in the net and the set of output actions in another one. The set $\text{envInput}_N$ (envOutput$_N$) corresponds to the set of not shared input (output) actions appearing in $\mathcal{N}$, that is, the input (output) actions labelling external transitions.

**Example 3.5** Let us consider the NETCOM depicted in Figure 4. It can be seen as an evolution of Example 3.3 in which we have included communication channels. In this case, the image pre-processing, previously performed by a single FSM, has been split.

---

**Figure 4.** Specification of pre-processing and checking image phases by using the CFSM model.
between two CFSM, pre-processing and checking. Moreover, we have included another CFSM, called database, that provides images to the image pre-processing system.

Next, we describe the steps to perform the pre-processing and checking phases required for the analysis of images. The database is in charge of providing the pre-processing node with an image stream (ImageRawo). The pre-processing node receives it (ImageRawi) and invokes the process which preprocesses the image (ImgNR). Once the pre-processed image is returned (ImgPP), the pre-processing node sends it (CImgPo) to the checking node (CImgPi) which invokes the process that checks the correctness of the generated image (CAP). Finally, when the verdict of the evaluation is received (YCP or NCP), the checking node sends it (ContPo or StopPo) to the pre-processing node. If the image presents any faults (StopPi), the pre-processing node stops the process and reports an error. If the image is correct (ContPo) the process can continue.

In order to validate the correctness of a system by using a passive testing technique, we record and analyse the sequences of actions generated by the system under test. These sequences are checked against a certain set of properties, that we call invariants, representing the most relevant properties that the system must fulfill. Next, we introduce the notion of communication invariant, an extension of the usual notion of invariant used in a single FSM.

**Definition 3.6** Let \( N = (M, C) \) be a NETCOM. We say that a sequence \( ? \) is a communicating invariant, in short c-invariant, for the net \( N \), if \( ? \) is defined according to the following EBNF:

\[
? := \vartheta_1 | \vartheta_2
\]

\[
\vartheta_1 := \varphi | \vartheta_2 \mid \varphi \mid \vartheta_3 \mid \varphi \rightarrow S
\]

\[
\vartheta_2 := \varphi \mid \vartheta_1 \mid \varphi \mid \vartheta_3 \mid \varphi \rightarrow O
\]

\[
\vartheta_3 := \star
\]

where \( \varphi \in \text{shared}_N \), \( \varphi \in \text{envInput}_N \), \( \varphi \in \text{envOutput}_N \), \( S \subseteq \text{shared}_N \) and \( O \subseteq \text{envOutput}_N \). The set of invariants for the net \( N \) is denoted by \( \Omega_N \), where we will omit the subindex if it can be deduced from the context.

The previous EBNF expresses that a c-invariant is a sequence of symbols where each component, but the last one, is either a pair with one of the elements being a shared action (s) and the other one an input (i) or an output (o) action, or the wildcard \( \star \) that can replace a sequence of actions not containing the first input symbol that appears in the component of the c-invariant that follows it. Let us note that two consecutive pairs in the sequence \( a/b, c/d \) must be compatible, that is, either \( c = b \in \text{shared}_N \) or \( b \in \text{envOutput}_N, c \in \text{envInput}_N \) and both \( a \) and \( c \) belong to the set of input actions of the same CFSM in \( N \). In addition, a c-invariant cannot contain two consecutive occurrences of \( \star \). The last component is given by either the expression \( i \rightarrow S \) or \( s \rightarrow O \). The former corresponds to an input action followed by a set of shared actions and the latter represents a shared action followed by a set of output actions.

A c-invariant contains two different components. The first one, called preface, includes a sequence of pairs of input/output actions in which one of them must correspond to a communication action between two machines. The second component represents the behaviour that the system must exhibit if we observe the sequence of communicating actions expressed in the preface. If we observe the preface but the next action produced
by the system is not included in the last component of the c-invariant then an error will have been detected.

4. Distributed framework to detect suspicious artefacts

In this section we describe our proposed distributed framework, called FORTIFIER, for detecting suspicious artefacts. In order to show a detailed perspective of our approach, a formal specification of the different phases of this framework is provided. We also present a set of c-invariants for analysing the efficiency and effectiveness of our proposal.

Figures 5 and 6 show the formal specification of the NETCOM that represents the behaviour of FORTIFIER. This specification has been developed using the formalism described in Section 3. It represents the different steps that constitute the whole process used for detecting suspicious artefacts that have been implemented in FORTIFIER. We can distinguish three different phases. The first one corresponds to image pre-processing operations, the second one ensures image integrity and the third one performs image recognition based on majority voting process. First, an image to be processed (ImgRawi) is sent to the pre-processing node. The image is filtered and processed (ImgNR) with noise reduction algorithms to fix possible visual defects of the image. Next, the pre-processed image (ImgPP) is sent to the checking node (CImgPo), where the image (CImgPi) is received and checked in order to detect format defects (CI).
some fault is detected, the checking node reports an error (StopPo) and the execution is aborted (EImgo). If the image is correct, the system sent it to the detection voting node (VotImgo). The voting process is performed for determining the suspect nature of an element. The image received (VotImg) is sent to the detection algorithms that process it (V1o, V2o, V3o). If the algorithm detects that the image matches with a suspicious artifact, it emits a positive vote (YA1). On the contrary, it emits a negative vote (NA1). Finally, if the majority of the votes is positive, an alarm is triggered (WD1) and the image is stored into a database (SaveDB).

In FORTIFIER, an image stream flows through the different stages by following a pipeline model, where the output generated by a node is the input of the one that performs the next step. Although each node only can process one image at the same time, the processing of different images can be performed simultaneously. Moreover, since the distributed design of FORTIFIER allows to execute the nodes in different physical machines, the resources provided by a distributed system, like an HPC cluster or a cloud computing system, can be exploited in parallel to increase the overall system performance.

We consider that the detection of elements that can be a threat to the security is critical. Thus, we provide a robust and extensible meta-detector suitable for different hazard environments, such that the detection of threatening elements is performed through a voting process in which participate several independent detection algorithms. In addition, we have included into FORTIFIER Online Stage (Mery, 2015b), a detection algorithm based on the classification and analysis of the main key points detected in an image.

Figures 6 and 7 show the specification of FORTIFIER, that is represented by a NETCOM with three CFSM. The first one, Online State, describes the general behaviour of the
algorithm, receiving an image ($V_1$, $V_2$, or $V_3$) and distributing tasks among the other machines ($M_{Ao}$, $M_{Vo}$). The second machine represents the behaviour of the Monocular analysis process, that is in charge of performing operations such as segmentation (SEG), key points selection (KEY), classification and clustering (CCS). The third machine, Multiple View Analysis, performs operations such as data association (DAS) and data analysis (DAN).

A positive verdict ($YA_1$) will be emitted by the algorithm if a suspicious artefact has been detected.

Next, we introduce a set of c-invariants. They represent behaviours that must be fulfilled by the system in order to ensure its correctness.

$$\theta_1 = \frac{NCP \rightarrow \{Stop\}_s}{i}$$

Intuitively, this c-invariant expresses that if the checking node detects any fault in an image (NCP), the system stops the process (Stop) of analysis of this image.

$$\theta_2 = \frac{Img_{Raw} / Img_{NR}, \star, NDAS / NMV_\star, NCCS / NMA_\star, \star, NWD_\star \rightarrow \{END\}}{s/o \ \ i/s \ \ i/s \ \ s \ \ O}$$

The above c-invariant expresses that after an image is loaded for being processed (Img_{Raw} / Img_{NR}), if we observe that no errors are detected during the multiple view analysis and the monocular analysis of two different voting algorithms, then the process should finish (END).

$$\theta_3 = \frac{ContP / Vote\_Img_\star, \star, MA_\star / SEG \star, YCCS \rightarrow \{YMA_\star\}}{i/s \ \ s/o \ \ s \ \ \ i \ \ \ \ s \ \ \ \ \ \ O}$$

The $\theta_3$ c-invariant represents that after an image has been sent to the detection node for voting (ContP / Vote\_Img), if at some point the monocular analysis process detects a suspicious artefact (YCCS) then it must be notified (YMA).

$$\theta_4 = \frac{Img_{PP} / C\_Img_\star, \star, ContP / Vot\_Img_\star, \star, NA_\star / NW_\star \ NW_\star \rightarrow \{END\}}{i/s, \star, i/s \ \ i/s \ \ s \ \ O}$$

The last c-invariant establishes that if an image is successfully pre-processed and checked (Img_{PP} / C\_Img) and the image is transferred to the detection voting, then, if the vote of
the third participant is negative, the verdict of the detection voting will be negative and the process ends.

5. Experiments

In this section we present several experiments to evaluate the scalability of FORTIFIER when it is deployed in different cloud systems. In this case, each cloud system consists of a single data-centre, which have been modelled by using the SIMCAN simulation platform (Núnez et al., 2012). Figure 8 shows the deployment infrastructure used for these experiments. In this configuration we can differentiate two main parts. First, a centralized data base that contains the images to be processed. Second, different cloud systems that access the shared data base through the Internet using a communication network.

Each cloud system contains one or several instances of FORTIFIER, which are executed for processing the corresponding set of images that are allocated in the centralized data base. It is important to remark that each instance of FORTIFIER is totally independent. For instance, two different processes that represent the same phase, like pre-processing and checking, of different FORTIFIER instances are also treated as different processes in the simulated environment. Also, each one of these processes are executed in a dedicated CPU core. Consequently, the number of FORTIFIER instances executed in the same cloud depends of the number of CPU cores used in each physical machine. Using this configuration we increase the level of parallelism in two levels. First, intra-cloud parallelism is obtained when different processes of FORTIFIER are executed in parallel using

Figure 8. Deployment infrastructure of FORTIFIER.
different physical machines of the same cloud system. Second, inter-cloud parallelism is obtained when different instances of **FORTIFIER** are executed in parallel using several cloud systems.

It is important to note that the data representing the repository of images in the database have been randomly generated. Thus, this data has been used to create simulated scenarios where our approach has been developed. In order to analyse the scalability of our proposed framework, **FORTIFIER** has been deployed in different scenarios containing 1, 2, 4 and 8 homogeneous cloud systems. It is worth to mention that all clouds have the following hardware configuration:

- CPU processor containing 1, 2, 4 and 8 CPU cores
- Hard disk drive of 1 TB.
- Network: Ethernet 100 Mbps, Ethernet Gigabit, Ethernet 10 Gbps and Ethernet 100 Gbps.
- 16 GB of RAM memory.

Initially, the system depicted in Figure 8 containing one cloud system has been modelled. In this experiment, one data-centre has been modelled using different configurations, such as the communication network and the number of CPU cores included in each physical machine. Figure 9 shows the obtained results, where the x-axis shows the type of communication network and the y-axis shows the obtained throughput, measured in processed images per minute. This chart depicts that both the communication network and the computing system act as a system bottleneck. First, the database is shared by all the physical machines and, therefore, all the images are transmitted through the same channel, which significantly decreases the overall system performance. Second, increasing the number of CPU cores per physical machines also increases the number of **FORTIFIER**

![Figure 9. Performance of **FORTIFIER** executed in a single cloud.](image-url)
instances executed in parallel. Hence, the level of parallelism for processing images is increased, which has a direct impact in the overall performance. Consequently, the best performance is obtained when both CPU resources and network bandwidth are increased.

In order to analyse the scalability of FORTIFIER when it is deployed in several cloud systems, we have modelled a scenario containing 1, 2, 4 and 8 clouds, each cloud having the same hardware configuration. This scenario uses an Ethernet 10 Gbps network. Figure 10 shows the overall performance when FORTIFIER is deployed in a multi-cloud environment, where the x-axis represents the number of clouds, the z-axis represents the number of CPU cores per physical machine and the y-axis represents the obtained throughput measured in processed images per minute.

In this case, increasing the number of clouds where FORTIFIER is deployed has a direct impact in the overall system performance, leading to a performance speed-up. However, increasing the number of CPU cores per machine slightly increases the system performance. This is mainly caused by the bottleneck located in the database system, which hampers the exploitation of computing parallelism by using all the CPU cores at the same time.

6. Conclusions and future work

In this paper we have presented FORTIFIER, a formally specified and analysed distributed framework, to detect suspicious artefacts. FORTIFIER has been specified by using a formal framework based on Finite State Machines. Also, a set of communicating requirements to check the correct behaviour of the proposed framework has been provided. In order to show the applicability of FORTIFIER it has been deployed along several cloud systems in a simulated environment. The experiments of this paper have been conducted by using the SIMCAN simulation platform. The evaluation results show that FORTIFIER provides an increasing in the overall system performance when it is deployed in
different cloud systems. However, since all the images are stored in a centralized data base, the communication network to access the data base acts as a bottleneck, which leads to a performance loss.

A first line of future work consists in the inclusion of timed and probabilistic information in our models (Andrés, Merayo, & Núñez, 2012; Hierons & Merayo, 2009; Hierons, Merayo, & Núñez, 2009). We would also like to use passive testing techniques to check the proposed framework by using more complex communicating requirements. A third line of work consists in increasing performance due to parallelization (Núñez, Filgueira, & Merayo, 2013; Núñez & Merayo, 2014). Finally, we would like to use learning techniques to improve the performance of our detection algorithms taking into account that an attacker might modify some of the components (López, Núñez, Rodríguez, & Rubio, 2002).

**Acknowledgments**

We would like to thank Manuel Núñez for useful discussions during the preparation of this paper.

**Disclosure statement**

No potential conflict of interest was reported by the authors.

**Funding**

Research partially supported by the Spanish Ministerio de Economía y Competitividad projects ESTuDio and DArDOS [TIN2012-36812-C02-01 and TIN2015-65845-C3-1-R] and the Consejería de Educación, Juventud y Deporte, Comunidad de Madrid project SICOMORo-CM [S2013/ICE-3006].

**Notes on contributors**

**Pablo C. Cañizares** is a PhD Student who received an M.Sc. degree in Computer Science in 2015 from the Universidad Complutense de Madrid. He has published 10 papers in refereed journals and international venues. His research interests are formal methods, modelling and testing of distributed systems.

**Mercedes G. Merayo** received her Ph.D. in Computer Science from Universidad Complutense de Madrid, Spain, in 2009. She holds an Associate Professor position in the Computer Systems and Computation Department at the same University. She has published more than 40 papers in refereed journals and international venues. Her research interests are formal methods in general, with a focus on probabilistic/timed/stochastic extensions in formal testing.

**Alberto Núñez** received an M.Sc. degree in Computer Science in 2005 from the Universidad Carlos III de Madrid, and a Ph.D. degree in Computer Science in 2011 from the same university. He is currently Teaching Assistant at the Universidad Complutense de Madrid, teaching Distributed Systems, Discrete Mathematics and Data Bases. He won the IBM Ph.D. Fellowship award in 2009. His research interests are focused on formal testing and high performance computing architectures and applications, especially on how to perform models and simulations.

**ORCID**

P. C. Cañizares [http://orcid.org/0000-0002-2084-1558](http://orcid.org/0000-0002-2084-1558)

M. G. Merayo [http://orcid.org/0000-0002-4634-4082](http://orcid.org/0000-0002-4634-4082)

A. Núñez [http://orcid.org/0000-0001-8613-746X](http://orcid.org/0000-0001-8613-746X)
References

107-71, U. P. L. (2001). *Aviation and transportation security act*. Washington, DC: United States Government Publishing Office.

Al-Qubaa, A., & Tian, G. (2012). Weapon detection and classification based on time-frequency analysis of electromagnetic transient images. *International Journal on Advances in Systems and Measurements*, 5(3), 89–99.

Andrés, C., Merayo, M. G., & Núñez, M. (2012). Formal passive testing of timed systems: Theory and tools. *Software Testing, Verification and Reliability*, 22(6), 365–405.

Baştan, M. (2015). Multi-view object detection in dual-energy X-ray images. *Machine Vision and Applications*, 26(7–8), 1045–1060.

Cañizares, P. C., Merayo, M. G., & Núñez, A. (2016). FARTHEST: FormAl distRibuTed scHema to dEtect Suspicious arTefacts. In *8th Asian conference on intelligent information and database systems, ACIIDS’16, LNAI 9621* (pp. 770–779). Da Nang: Springer.

Cavalli, A. R., Higashino, T., & Núñez, M. (2015). A survey on formal active and passive testing with applications to the cloud. *Annales of Telecommunications*, 70(3–4), 85–93.

Franzel, T., Schmidt, U., & Roth, S. (2012). Object detection in multi-view X-ray images. *Pattern Recognition*, 7476, 144–154.

Gaudel, M.-C. (1995). Testing can be formal, too! In *6th int. joint conf. CAAP/FASE, theory and practice of software development, TAPSOFT’95, LNCS 915* (pp. 82–96). Aarhus: Springer.

Grieskamp, W., Kicillof, N., Stobie, K., & Braberman, V. (2011). Model-based quality assurance of protocol documentation: Tools and methodology. *Software Testing, Verification and Reliability*, 21(1), 55–71.

Hierons, R. M., Bogdanov, K., Bowen, J., Cleaveland, R., Derrick, J., Dick, J., … Zedan, H. (2009). Using formal specifications to support testing. *ACM Computing Surveys*, 41(2), 9.

Hierons, R. M., Bowen, J., & Harman, M. (Eds.) (2008). *Formal methods and testing, LNCS 4949*. Berlin: Springer.

Hierons, R. M., Merayo, M. G., & Núñez, M. (2016). Controllability through nondeterminism in distributed testing. In *28th IFIP WG 6.1 int. conf. on testing software and systems, ICTSS’16, LNCS 9976* (pp. 89–105). Graz, Austria.

Hierons, R. M., Merayo, M. G., & Núñez, M. (2017). An extended framework for passive asynchronous testing. *Journal of Logic and Algebraic Programming*, 86(1), 408–424.

Liu, D., & Wang, Z. (2007). A method of X-ray image recognition based on fuzzy rule and parallel neural networks. In *International symposium on neural networks* (pp. 1231–1239). Nanjing: Springer.

Liu, D., & Wang, Z. (2008). A united classification system of X-ray image based on fuzzy rule and neural networks. In *3rd int. conf. on intelligent system and knowledge engineering, ISKE’08* (Vol. 1, pp. 717–722). Xiamen: IEEE Computer Society.

López, N., Núñez, M., Rodríguez, I., & Rubio, F. (2002). Including malicious agents into a collaborative learning environment. In *8th int. conf. on intelligent tutoring systems, LNCS 2363* (pp. 51–60). Jhongli: Springer.

Merayo, M. G., & Núñez, A. (2015). Passive testing of communicating systems with timeouts. *Information and Software Technology*, 64, 19–35.

Mery, D. (2013). X-ray testing: The state of the art. *Journal Department of Computer Science-Pontificia Universidad Catolica de Chile Av. Vicuna Mackenna. Santiago de Chile*, 18(9), 1–12.

Mery, D. (2014). Computer vision technology for X-ray testing. *Insight-Non-Destructive Testing and Condition Monitoring*, 56(3), 147–155.

Mery, D. (2015a). Applications in X-ray testing. In *Computer vision for X-ray testing* (pp. 267–325). Berlin: Springer.
Mery, D. (2015b). Inspection of complex objects using multiple-X-ray views. *IEEE/ASME Transactions on Mechatronics*, 20(1), 338–347.

Mery, D., Riffo, V., Zuccar, I., & Pleringer, C. (2017). Object recognition in X-ray testing using an efficient search algorithm in multiple views. *Insight-Non-Destructive Testing and Condition Monitoring*, 59(2), 85–92.

Murray, N. C., & Riordan, K. (1995). Evaluation of automatic explosive detection systems. In 29th annual int. Carnahan conf. on security technology (pp. 175–179). Surrey: Institute of Electrical and Electronics Engineers.

Nercessian, S., Panetta, K., & Agaian, S. (2008). Automatic detection of potential threat objects in X-ray luggage scan images. In *IEEE conference on technologies for homeland security* (pp. 504–509). Waltham: IEEE Computer Society.

Núñez, A., Fernández, J., Filgueira, R., García, F., & Carretero, J. (2012). SIMCAN: A flexible, scalable and expandable simulation platform for modelling and simulating distributed architectures and applications. *Simulation Modelling Practice and Theory*, 20(1), 12–32.

Núñez, A., Filgueira, R., & Merayo, M. G. (2013). SANComSim: A scalable, adaptive and non-intrusive framework to optimize performance in computational science applications. In *13th int. conf. on computational science, ICCS’13, procedia computer science* 18 (pp. 230–239). Barcelona: Elsevier.

Núñez, A., & Merayo, M. G. (2014). A formal framework to analyze cost and performance in Map-Reduce based applications. *Journal of Computational Science*, 5(2), 106–118.

Paranjape, R., Sluser, M., & Runtz, E. (1998). Segmentation of handguns in dual energy X-ray imagery of passenger carry-on baggage. In *Canadian conference on electrical and computer engineering* (Vol. 1, pp. 377–380). Waterloo: IEEE.

Singh, S., & Singh, M. (2003). Explosives detection systems (eds) for aviation security. *Journal of Signal Processing*, 83(1), 31–55.

Singh, M., & Singh, S. (2004). Image segmentation optimisation for X-ray images of airline luggage. In *Proceedings of the international conference on computational intelligence for homeland security and personal safety, CIHSPS’04* (pp. 10–17). Venice: IEEE.

Turcsany, D., Mouton, A., & Breckon, T. (2013). Improving feature-based object recognition for X-ray baggage security screening using primed visual words. In *International conference on industrial technology (ICIT’13)* (pp. 1140–1145). Cape Town: IEEE.

Uroukov, I., & Speller, R. (2015). A preliminary approach to intelligent X-ray imaging for baggage inspection at airports. *Signal Processing Research*, 4, 1–11.

Veanes, M., Campbell, C., Grieskamp, W., Schulte, W., Tillmann, N., & Nachmanson, L. (2008). Model-based testing of object-oriented reactive systems with Spec Explorer. In *Formal methods and testing, LNCS 4949* (pp. 39–76). Berlin: Springer.

Wells, K., & Bradley, D. A. (2012). A review of X-ray explosives detection techniques for checked baggage. *Applied Radiation and Isotopes*, 70(8), 1729–1746.