# Anomaly Detection In Secure Cloud Environments Using a Self-Organizing Feature Map (SOFM) Model For Clustering Sets of R-Ordered Vector-Structured Features

Ioannis M. Stephanakis
OTE (Hellenic Telecom Organization)
99 Kifissias Avenue
GR-151 24, Athens, Greece
+30 210 6115798
stephan@ote.gr

Ioannis P. Chochliouros
Research Programs Section, OTE
Pelika & Spartis Str.
GR-151 22 Maroussi, Greece
+30 210-6114651
ichochliouros@oteresearch.gr

Evangelos Sfakianakis
Research Programs Section, OTE
Pelika & Spartis Str.
GR-151 22 Maroussi, Greece
+30 210-6114938
esfak@oteresearch.gr

Noorulhassan Shirazi
School of Comp. & Communications
Lancaster University, UK
+44 1524 510380
n.shirazi@lancaster.ac.uk

## ABSTRACT

Cloud computing delivers services over virtualized networks to many end-users. Cloud services are characterized by such attributes as on-demand self-service, broad network access, resource pooling, rapid and elastic resource provisioning and metered services of various qualities. Cloud networks provide data as well as multimedia and video services. Cloud computing for critical structure IT is a relative new area of potential applications. Cloud networks are classified into private cloud networks, public cloud networks and hybrid cloud networks. Anomaly detection systems are defined as a branch of intrusion detection systems that deal with identifying anomalous events with respect to normal system behavior. A novel application of a Self-Organizing-Feature Map (SOFM) of reduced/aggregate sets of ordered vector structured features that are used for detecting anomalies in the context of secure cloud environments is herein proposed. Multivalue inputs consist of reduced/aggregate ordered sets of vector and binary features. The nodes of the SOFM - after training - are indicative of local distributions of feature measurements during normal cloud operation. Anomalies are detected as outliers of the trained SOFM. Each structured vector consists of binary as well as histogram data. The aggregated Canberra distance is used to order histogram data whereas the Jaccard distance is used for multivalue binary data. The so-called *Cross-Order Distance Matrix* is defined for both cases. The distance depends upon the selection of a similarity/distance *measure* and a *method* for operating upon the elements of the *Cross-Order Distance Matrix*. Several methods of estimating the distance between two ordered sets of features are investigated in the course of this paper.

## Categories and Subject Descriptors

• **Networks~Network properties.** Network security.
• **Networks~Distributed architectures.** Cloud computing.
• **Theory of computation~Design and analysis of algorithms.** Parallel algorithms: Self-organization.


*16th EANN workshops,* September 25 - 28, 2015, Rhodes Island, Greece
 ISBN 978-1-4503-3580-5/15/09…$15.00
DOI: http://dx.doi.org/10.1145/2797143.2797145

## Keywords

Secure cloud networks; intrusion detection; Reduced/aggregate-ordering; Self-Organizing Feature Maps (SOFMs), Canberra distance; Jaccard distance; clustering.

## 1. INTRODUCTION

Cloud computing delivers computing services from large, highly virtualized network environments to many independent users, using shared applications and pooled resources. One may distinguish between *Software-as-a-Service* (SaaS), in which case software is offered on-demand through the internet by the provider and it is parameterized remotely (like for example on-line word processors, spreadsheets, Google Docs and others), *Platform-as-a-Service* (PaaS), in which case customers are allowed to create new applications that are remotely managed and parameterized and the platform offers tools for development and computer interface restructuring (like for example Force, Google App Engine and Microsoft Azure) and *Infrastructure-as-a-Service* (IaaS), in which case virtual machines, computers and operating systems may be controlled and parameterized remotely (like for example Amazon EC2 and S3, Terremark Enterprise Cloud, Windows Live Skydrive, Rackspace Cloud, GoGrid, Joyent, AppNexus and others). Cloud computing may be divided into *public cloud*, where everyone may register and use the services, *private cloud* - that is accessible through a private network - and *partner cloud* - that offers services to specific partners/users. A hybrid cloud is a combination of private/internal and external cloud resources that enables outsourcing of noncritical functions whilst keeping the remainder internal. The key functionality is the ability to use and release resources from public clouds as and when required. This is used to handle sudden demand surges ("*flash crowds*") and is known as "*cloud-bursting*". Cloud computing is an on-demand service whose size depends upon users' needs and should feature scale flexibility. It is built upon such network elements as switches supporting novel communication protocols, specific servers based on *Virtual Machine* (VM) technology and dynamic resource management as well as *Network-Attached-Storage* (NAS). Specific software platforms may be used for service orchestration in cloud environments (like for example OpenStack).

Modern virtualized cloud environments promote the aspects of elasticity and resource transparency that are enabled by service and *Virtual Machine* (VM) migration. The majority of cloud management software ensures that a VM retains its network identity and connectivity by initiating a "hot" or "line" migration strategy. "Hot" migration allows to move the entire running VM (i.e. its active memory and execution state) from one physical node of the cloud to another without significant downtime as an effective newline resource-management strategy empowering workload balancing. On the other hand, a VM should be powered off before migration during the so called "cold" migration. Network anomaly detection in the cloud is a challenging area of active research and several software tools have been developed to this end [1], [2]. Anomaly detection systems are defined as a branch of intrusion detection systems that deal with identifying anomalous events with respect to normal system behavior. Such systems assume a model of normal behavior and issue alerts whenever operational characteristics deviate from the prescribed "normal" behavior making a suitable assumption that such changes are frequently caused by malicious or disruptive events. Anomaly detection techniques for cloud environments are still evolving due to the fact that the topic presents several challenging aspects.

Anomalies are classified as point anomalies – if an individual data instance can be considered as anomalous with respect to the rest of the data – contextual anomalies – if an information occurrence is anomalous in a precise context – and collective anomalies – if a collection of data instances are anomalous with respect to the entire data set. **Table 1** summarizes common anomaly types [3]:

**Table 1. Common anomalies**

| Anomaly | Definition |
|---|---|
| ALPHA | Unusually high rate point-to-point byte transfer |
| DOS, DDOS | (Distributed) Denial of service attack against a single victim |
| FLASH CROWD | Unusually large demand for resource/service emerging from common set of sources |
| SCAN | Scanning a host for a vulnerable point (port scan) or scanning the network for a target port (network scan) |
| WORM | Self-propagating code that spreads across a network |
| POINT to MULTIPONT | Distribution of content from one server to many servers |
| OUTAGE | Equipment related events that decrease traffic exchange by an Origin-Destination pair |
| INGRESS-SHIFT | Customer shifts traffic from one ingress point to another |

Detection techniques of system anomalies include Principal Component Analysis (PCA) [4],[5], clustering-based methods [5],[6], Naive Bayesian approaches [7] and Expectation-Maximization Gaussian Mixture Models [8]. Self-Organizing Maps (SOM) have been used for intrusion detection as well (see for example, [9]). Ordered sequences, i.e. continuous and discontinuous pattern matching, constitute an alternative proposition [10]. A survey of anomaly detection approaches is given in [11],[12],[13]. Several approaches for anomaly detection have been tested within the framework of the SECCRIT project [3],[14] see **Fig. 1** .

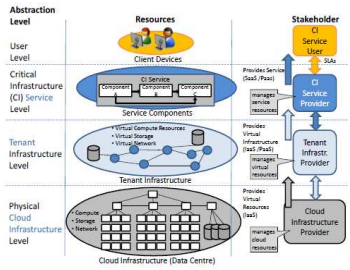


**Figure 1. The SECCRIT architectural framework**

The contribution of this paper consists mainly of presenting a novel approach of producing *Self-Organizing Feature Maps* (SOFMs) of sets of ordered structures. The structures within a set may contain binary as well as vector components. Specific measures are proposed for each case. **Section 2.1** outlines the notions of *Reduced* (*aggregate*) *ordering sets* whereas **Section 2.2** describes distance measures based on the so called *Cross-Order Distance matrix*. The details of the proposed algorithm are given in **Section 3** and numerical simulations for structured measurements pertaining to anomaly detection in cloud environment are described in **Section 4**. This work concluded by commenting upon the accuracy of the method and suggestions of further research.

# 2. REDUCED / AGGREGATE - ORDERING AND SELF-ORGANIZING FEATURE MAPS (SOFMs)

## 2.1 Background and Definitions

There is no universally accepted method for ordering multivariate data. Widely known multivariate ordering methods include [15]:

- *Marginal ordering* (*M-ordering*) according to which feature vectors are ordered in each component independently. This scheme produces a set of ordered output vectors that is usually not the same as the set of input vectors.
- *Conditional ordering* (*C-ordering*) according to which vectors are ordered based on the marginal ordering of one of their components. This scheme disregards the vectorial nature of the multichannel data.
- *Partial ordering* (*P-ordering*) according to which vectors are partitioned into smaller groups that are then ordered. Despite its theoretical appeal, this scheme is computationally demanding. Since partial ordering is difficult to perform in more than two dimensions, it is not appropriate for three-component signals.
- *Reduced* (*aggregate*) *ordering* (*R-ordering*) according to which the feature vectors are first reduced to scalar representatives using a suitable distance or similarity measure. The ordering of these scalars is then taken as the ordering of the corresponding vectors. This is the most common ordering scheme in the literature.

The reduced ordering scheme is the most attractive and widely used in signal processing since it relies an overall ranking of the original set of input samples and the output is selected from the same set. The ordered sequence of scalar values $D(1) \le D(2) \cdots \le D(i) \le \cdots D(N)$ for $i = 1,2 \ldots N$ implies the

same ordering of the corresponding vectors $\mathbf{x}_i$ , i.e. {**x** (1), **x** (2) … **x** (*i*) … **x** (*N*)}. *R*-ordering non-linear processing is based on the ordering of aggregated distances, i.e. $D_i = \sum_{j=1}^{N} d(\mathbf{x}_i, \mathbf{x}_j)$ or aggregated similarities $D_i = \sum_{j=1}^{N} similarity(\mathbf{x}_i, \mathbf{x}_j)$ [16], [17] (see for example [18]).

Sampling of binary and vector features is carried out over all host and client servers connected to the cloud, which are denoted as $i, j \in \{A, B, C, D....\}$ for a time window [$t_1$, $t_2$] according to **Fig. 2**. Hence a ranking of all host and client servers connected to the cloud results after aggregate ordering of their feature vectors as explained in the previous paragraph. Spreading feature vectors over a considerable distance range is indicative of an anomaly. Ordered sequences of feature vectors during normal cloud operation are clustered in nodes using a SOFM.

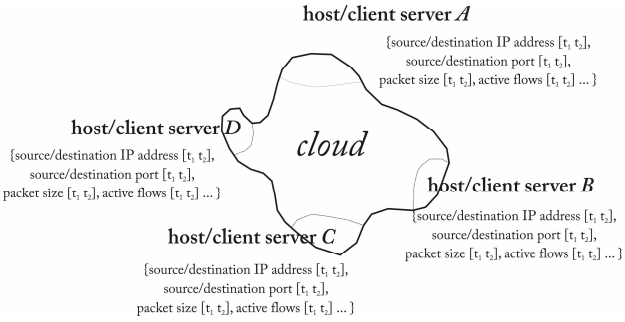


**Figure 2. Sampling binary and vector features in the cloud**

## 2.2 Definition of the Cross-Order Distance Matrix Between Ordered Vectors and Binary Data

The *Cross-Order Distance Matrix* is defined along with a distance or similarity measure and a method of selecting the elements of the *Cross-Order Distance Matrix* or operating upon them in order to estimate the distance between two ordered sets of feature vectors, denoted as *S*={**x** (1), **x** (2) … **x** (*i*) … **x** (*N*)} and *S*'={**x'** (1), **x'**(2) … **x'** (*i*) … **x'** (*N*)}. Thus,

$$\mathbf{D}_{SS'} = [dist(\mathrm{x}_i, \mathrm{x}_j) \text{ where } i, j = 1...N; measure, method] = \begin{pmatrix} d(x^S(1), x^{S'}(1)) & d(x^S(1), x_s^{S'}(2)) & \cdots & d(x^S(1), x^{S'}(N)) \\ d(x^S(2), x^{S'}(1)) & d(x^S(2), x^{S'}(2)) & & d(x^S(2), x^{S'}(N)) \\ \vdots & & \ddots & \\ d(x^S(N), x^{S'}(1)) & d(x^S(N), x^{S'}(2)) & \cdots & d(x^S(N), x^{S'}(N)) \end{pmatrix} \quad (1)$$

A *method* can be the sum of all elements of the *Cross-Order Distance Matrix*, its trace (defined as $trace\{\mathbf{D}_{SS'}\} = \sum_{k=1}^{N} d(x^S(k), x^{S'}(k))$ ), constant thresholding of all elements of the matrix (i.e. setting all elements below the threshold equal to zero and carry out summation over all non-zero elements), non-constant threholding using a rule such as

$$threshold = \min(d(x_1^S, x_N^S),\ d(x_1^{S'}, x_N^{S'})) \quad (2)$$

or the dispersion of distances over permutations of feature vectors between the two ordered sets (i.e. $\underset{all\ s'_1 s_1}{averege}\{\mathbf{D}_{s'_1 s_1}\}$ where $S_{S',1}$={**x'**(1), **x** (2) … **x** (*i*) … **x** (*N*)}, {**x**(1), **x'**(2) … **x** (*i*) … **x** (*N*)} … {**x**(1), **x** (2) … **x** (*i*) … **x'**(*N*)} etc.

Let us consider sets of data that are indicative of the state of the cloud within time interval [$t_1$ $t_2$], i.e. $x_s(f_{\text{source/destinationIP}}, f_{\text{source/destination port}}.....;[t_1\ t_2])$ - where $s \in \{A, B, C, D....\}$ - and refers to some host/server connected to the cloud. Each node of the SOFM in **Fig. 3** consists an ordered set of samples {$x^S(1)$, $x^S(2)$, $x^S(3)$, $x^S(4)$ … } of feature vectors which corresponds to servers/hosts connected to the cloud. Distributions of measurements within interval [$t_1$ $t_2$] may refer to binary data - like for example IP addresses or ports - or scalar data, like for example packet sizes. Comparisons are carried out between ordered sets pertaining to interval [$t'_1$ $t'_2$], i.e. {$x^{S'}(1)$, $x^{S'}(2)$, $x^{S'}(3)$, $x^{S'}(4)$ … }, and ordered sets pertaining to interval [$t_1$ $t_2$], i.e. {$x^S(1)$, $x^S(2)$, $x^S(3)$, $x^S(4)$ … }. The *Cross-Order Distance Matrix* between two such ordered sets is defined in order to quantify the similarity-distance between them.

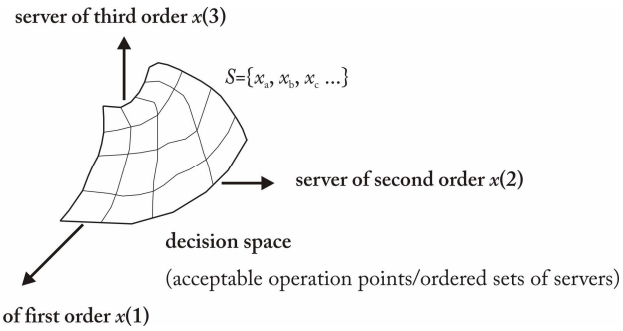


**Figure 3. SOFM for ordered sets - Each node consists of some ordered set of hosts**

Feature measurements during normal operation are clustered to the nodes of the SOFM according to some method applied upon the *Cross-Order Distance Matrix*. On the contrary measurements during an intrusion attack yield outliers of the trained SOFM. The proposed approach may use all or selected rank samples during training.

Ordering of the sample set is a necessary preprocessing step. We use the Canberra distance in order to find the distance between $x_{s'}$ and $x_s$ summing over all fields of the structured vectors of measurements, i.e.

$$d(x_{s'}, x_s; [t_1\ t_2]) = \{d(f_{\text{source/destinationIPaddress}}(s'), f_{\text{source/destinationIPaddress}}(s); [t_1\ t_2]) + d(f_{\text{source/destinationport}}(s'), f_{\text{source/destinationport}}(s); [t_1\ t_2]) + d(f_{\text{packetsize}}(s'), f_{\text{packetsize}}(s); [t_1\ t_2])\} = \left( \frac{|f_{\text{source/destinationIPaddress}}(s') - f_{\text{source/destinationIPaddress}}(s)|}{|f_{\text{source/destinationIPaddress}}(s')| + |f_{\text{source/destinationIPaddress}}|} + \frac{|f_{\text{source/destinationport}}(s') - f_{\text{source/destinationport}}(s)|}{|f_{\text{source/destinationport}}(s')| + |f_{\text{source/destinationport}}(s)|} + \frac{|f_{\text{packetsize}}(s') - f_{\text{packetsize}}(s)|}{|f_{\text{packetsize}}(s')| + |f_{\text{packetsize}}(s)|} \right) \quad (3)$$

One may construct the histogram over the lowest and the highest value of some feature in the training set in order to estimate the differences between such quantity variation within interval [$t_1$ $t_2$] for a specific host/server connected to the cloud. The differences between histograms are obtained using the Canberra distance over all histogram bins $\frac{1}{\#\ non\ zero\ bins} \sum_{l=1...L} \frac{|h^{s'}_{\text{packet size}}(bin_l) - h^{s}_{\text{packet size}}(bin_l)|}{|h^{s'}_{\text{packet size}}(bin_l) + h^{s}_{\text{packet size}}(bin_l)|}$. For binary data within [$t_1$ $t_2$] one may use the Jaccard distance in **Eq. 4**, which measures the dissimilarity between sample sets. It is

obtained by dividing the difference of the sizes of the union and the intersection of two sets by the size of the union.

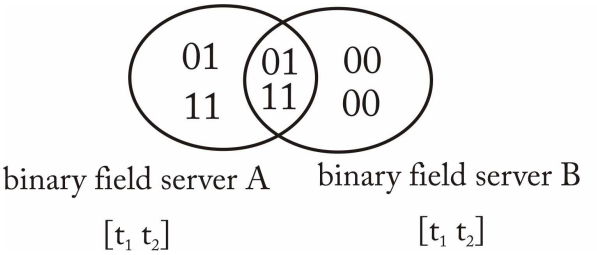


**Figure 4.a The Jaccard distance for binary data fields of vector measurements corresponding to host/client servers A and B for the interval $[t_1\ t_2]$**

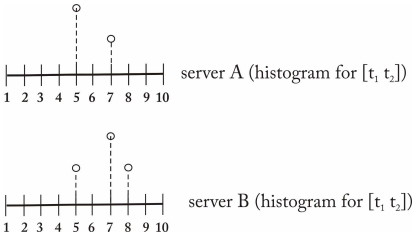


**Figure 4.b Difference of histograms corresponding to scalar fields of vector measurements for $[t_1\ t_2]$**

Canberra distance equals $\frac{1}{4}\sum_{all}\left(0\ \ 0\ \ 0\ \ 0\ \ \frac{1}{3}\ \ 1\ \ \frac{1}{3}\ \ 1\ \ 0\ \ 0\right)$

$$d_J(S,S') = 1 - J(S,S') = \frac{|S \cup S'| - |S \cap S'|}{|S \cup S'|} \qquad (4)$$

**Example:** Let one consider two sample vectors $x_A$ and $x_B$ pertaining to servers *A* and *B* for measurements over $[t_1\ t_4]$. Each sample vector consists of a binary and a scalar field, i.e.

$x_A = [f_b, f_s\,;\,[t_1, t_4]] =$

$= \{[\text{‘01’}, 5\,;\,t_1]\},\ \{[\text{‘01’}, 6\,;\,t_2]\}, \{[\text{‘11’}, 7\,;\,t_3]\}, \{[\text{‘11’}, 5\,;\,t_4]\}$

$x_B = [f_b, f_s\,;\,[t_1, t_4]] =$

$= \{[\text{‘00’}, 7\,;\,t_1]\},\ \{[\text{‘00’}, 8\,;\,t_2]\}, \{[\text{‘01’}, 5\,;\,t_3]\}, \{[\text{‘11’}, 7\,;\,t_4]\}.$

Hence the distance between $x_A$ and $x_B$ consisits of two parts (see **Fig. 4.a** and **4.b**), one derived from the binary field,

$d_J(f_b(x_A), f_b(x_A);\ [t_1,\ t_4])=2/3$ and one derived from the histogram of the scalar variable over $[t_1\ t_4]$, i.e. $d_{Canberra}(hist_A, hist_B;\ [t_1, t_4]) = =1/4(1/3+1+1/3+1)=2/3$.

Distances that are obtained using some process upon the *Cross-Order Distance Matrix* should allow for discerning between ordered sets. Two different ordered sets of eight feature vectors are used. One set consists entirely of samples of measurements taken during normal operations whereas the other consists of measurements taken during abnormal operation. **Fig. 5** illustrates two different *methods* for the same *Cross-Order Distance Matrix*. **Fig. 5.a** shows the upper and the lower limit of applying the *method* “*sum of all elements*” on *the Cross-Order Distance Matrix*. The rank elements from the ordered sets during abnormal operation are introduced in the *Cross-Order Distance Matrix* one-at-a-time by replacing columns or rows (value +8 corresponds to the *Auto-Order Distance Matrix* of the ordered set with anomaly measurements whereas value -8 corresponds to the *Auto-Order Distance Matrix* of the ordered set with measurements during normal operation). **Fig. 5.b** shows the upper and the lower limit of applying the *method* “0.1 *threshold*” on *the Cross-Order Distance Matrix*. Obviously thresholded distance matrix allows for better results should one consider clustering anomalies using a SOFM. We use the *method* “*sum of all elements*” during training since we cluster vectors of measurements pertaining to normal operation (i.e. the ordered set indicated by -8).

# 3. ANOMALY DETECTION USING SOFMs WITH MULTISET INPUTS

## 3.1 Outline of the Proposed Approach

The proposed approach consists of sampling the cloud network during abnormal conditions for small time windows $[t_1\ t_2]$, $[t_3\ t_4]$, $[t_5\ t_6]$ etc. and selecting samples of the form

$$x_s(f_{\text{packets}}, f_{\text{bytes}}, f_{\text{active flows}}, f_{\text{source/destinationIP}}, f_{\text{source/destination port}}, f_{\text{packet size}}; [t_1\ t_2])$$

for each server $s \in \{A, B, C.....\}$. The samples that correspond to host and client servers connected to the cloud are then ordered in ascending distance order according to the reduced ordering scheme described in **Section 2.1**. One may use selected members of the ordered set, like for example the first *K* members in order to train a SOFM as described in the sequel.

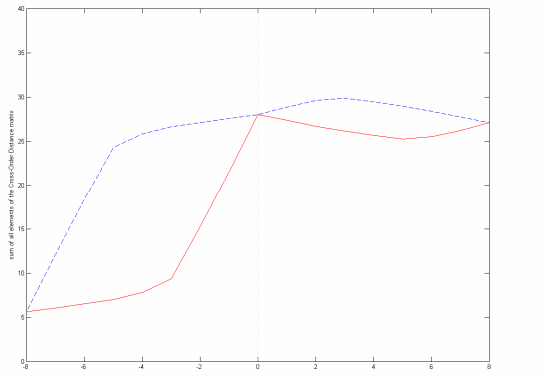

**Figure 5.a Distances defined for *method* “sum of all elements of cross-order distance matrix” (upper and lower bounds) vs number of substitutions of ordered elements from one set into the other**

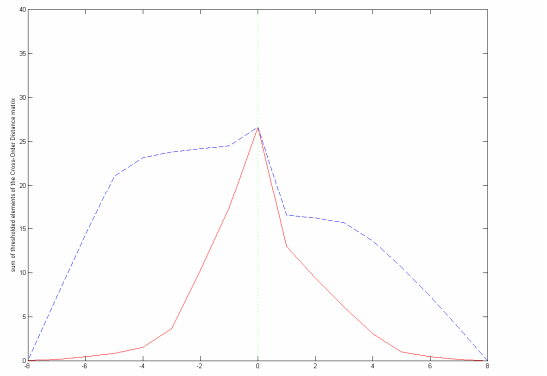

**Figure 5.b Distances defined for *method* “thresholded distance matrix - *threshold*=0.1” (upper and lower bounds) vs number of substitutions of ordered elements from one set into the other** (values -8 and +8 correspond to distance matrices from the same ordered set *measure*=Canberra)

## 3.2 Aggregating Multiset Inputs into Clusters – Training

There are three basic steps involved in the application of the SOFM algorithm after initialization, namely, sampling, similarity matching and updating. Reduced/aggregated ordering of sample structured vectors within time windows can be regarded as an intermediate step. The sum of the aggregated distances between fields of an ordered structure and a set of *K* feature vectors corresponding to a node of the SOFM is evaluated for all *L* nodes of the map. The *Cross-Order Distance Matrix* - as defined in **Section 2.2** - is used to derive the sum of the aggregated distances. The result of the application of the selected method upon the *Cross-Order Distance Matrix* is used to determine the winning neuron. The aforementioned steps are described in detail as follows:

1. *Initialization of the partial sets*. Choose the initial values for the weight vectors $w_j(0)$. Assume that each weight vector $w_j$ that corresponds to a neuron consists of a set of *K* representative host and client servers samples for the time window, i.e. $\mathbf{w}_j = \left(\mathbf{w}_{j,1}, \mathbf{w}_{j,2} \ldots \mathbf{w}_{j,K}\right)$ where index *j* equals 1, 2,…, *L* (where *L* stands for the total number of neurons).
2. *Sampling*. Sample cloud and server conditions for time window $[t_1\ t_2]$, $\mathbf{v}(t) = \left(x_A(t), \quad x_B(t), \quad x_C(t) \quad .....\right)$.
3. *Reduced/aggregated ordering* of the samples corresponding to the host/client servers connected to the cloud. Arrange vectors samples in a group $\mathbf{v}(t) = \left(x_1^{S(t)}, \quad x_2^{S(t)}, \quad x_3^{S(t)} \quad .....\right)$ in such a way that $D_k = \sum_{j=1,\, j\neq k}^{N} d(x^S(k), x^S(j)) \le D_l = \sum_{j=1,\, j\neq l}^{N} d(x^S(l), x^S(j))$ for $k < l$. Distance $d(x^S(l), x^S(j))$ equals $\sum_{all\ \alpha} d_a(f_\alpha(x^S(l)), f_a(x^S(j)))$ where
   $\alpha \in \{packets, bytes, active\ \ flows, source/destination\ IP\ address, source/destination\ port, packet\ size \ldots\}$
   .
4. *Similarity Matching*. Find the best-matching (winning) neuron $i(x^{S_1}, \quad x^{S_2}, \quad x^{S_3} \quad ..... \quad x^{S_L})$ at time *t* by aggregating the distances between the samples in the set and the *K* vectors at each of the *L* nodes, i.e.
   $$i(\mathbf{v}) = \arg\min_j (method\ (\mathbf{D}(\mathbf{w}_{i,k}, \mathbf{v}(t))))$$
   where $j = 1,2,..., L$ (5)
5. *Updating*. Adjust the synaptic weight vectors of all neurons, using the update formula
   $$\mathbf{w}_j(t+1) = \begin{cases} \mathbf{w}_j(t) + \eta(t)\left(\left(x_1^S(t) \quad x_2^S(t) \quad \cdots \quad x_k^S(t)\right) - \mathbf{w}_j(t)\right), & j \in \Lambda_{i(\mathbf{v})}(t) \\ \mathbf{w}_j(t), & \text{otherwise} \end{cases} \quad (6)$$
   where the ordered lowest *K* ranks of the training set are used, $\eta(t)$ is the learning-rate parameter and $\Lambda_{i(\mathbf{v})}(t)$ is the neighborhood function centered around the winning neuron. $\Lambda_{i(\mathbf{v})}(t)$ is varied dynamically during learning for best results.
6. *Continuation*. Continue with Step 2 until no noticeable changes in the feature map are observed.

## 3.3 Anomaly Detection

Samples pertaining to abnormal and normal network conditions are presented to the SOFM after training in order to detect anomalies. Similarity matching is carried out as described by **Step 4** of the algorithm summing all elements of a thresholded *Cross-Order Distance Matrix*. An anomaly is detected should the aggregate distance be higher than a threshold determined during training, i.e. anomalies are detected as outliers should the minimum distance from the nodes of the SOFM exceed a specified threshold. One assumes normal operation should minimum distance be lower than the threshold.

# 4. NUMERICAL SIMULATIONS

Data samples are taken from files with network measurements during normal and abnormal cloud operation. Several anomalies and introduced during VM migration like network and port-scan attacks and *Denial-of-Service* under heavy and light traffic conditions (see **Table 1**). *Denial-of-Service* attacks are realized using LOIC[1]. A typical ordering of six samples is given in the following figures. Ordered sets include a feature vector with anomaly measurements at *t*, where *t* ranges from 301 to 599, and five feature vectors corresponding to normal conditions ordered in ascending distance order. The feature vector ranked as sixth corresponds to anomalous conditions whereas the five first feature vectors correspond to normal conditions. The splitting of the histogram over distance in two parts is indicative of an anomaly (see **Figs. 6** for feature vectors corresponding to single time stamps).

It turned out that by taking a window of multiple time-stamps and extracting histograms for each feature separately increases the accuracy of the proposed method, i.e. the number of packets over ten (10) consecutive time stamps are used to construct a histogram of ten bins, the number of bytes are used to construct a histogram of twenty bins, the number of active flows are used to construct a histogram of fifteen bins, the entropies of source IP addresses are used to construct a histogram of twelve bins etc. (see **Fig. 7** and **Table 2**). Ordering multiple histogram samples using the Canberra distance in ascending order is given in **Figs. 7**. The top five histograms are indicative of normal network conditions whereas the lower histograms yield the higher distance and are indicative of an anomaly. The overall histogram over distance is split in two parts as expected (see **Figs. 8**).

A 10x10 SOFM is used in order to cluster ordered histogram samples. We consider sets of multiple feature vectors consisting of six different vectors selected at random. A Self-Organizing Feature Map (SOFM) is trained using six feature vectors corresponding to normal conditions ordered in ascending distance order. Six multiple histograms samples corresponding to normal network conditions (from *t* to *t*+9 where *t* ranges from 1 to 291 in **Fig. 9**) are ordered using the cumulative Canberra distance for all eight (8) feature histograms (see **Table 2**). A total of four hundred multiple histogram samples are used in order to train the SOFM. The sum of all elements of the *Cross-Order Distance Matrix* is used in order to train a SOFM using the Canberra distance (i.e. '*measure*'='*Canberra*' and '*method*'='*all*'). The number of epochs used to train the SOFM equals 120.

[1] LOIC is an open source network stress testing tool.

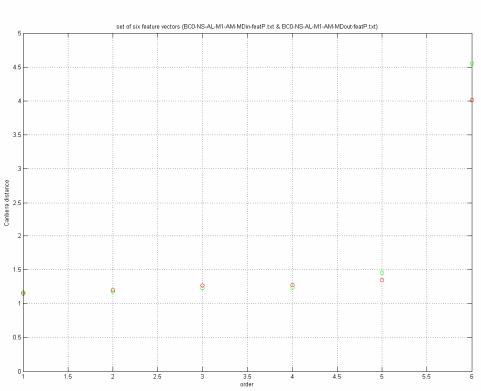

**Figure 6.a Distances vs rank of elements – 6th rank corresponds to anomaly** (single stamp feature vectors)

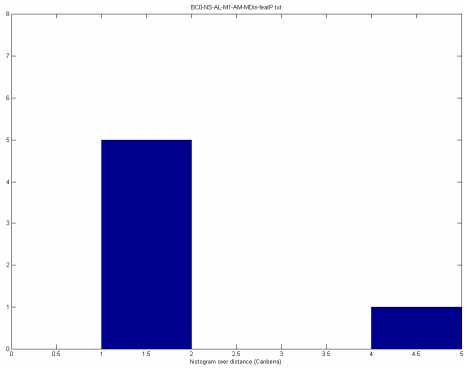

**Figure 6.b Number of ordered feature vectors over distance as a histogram – 6th rank corresponds to anomaly** (single stamp feature vectors-BC0-NS-AL-M1-AM-MDin)

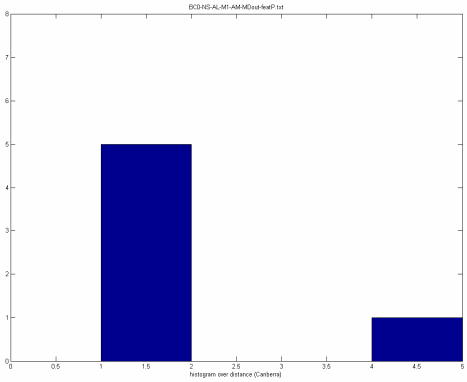

**Figure 6.c Number of ordered feature vectors over distance as a histogram – 6th rank corresponds to anomaly** (single stamp feature vectors-BC0-NS-AL-M1-AM-MDout)

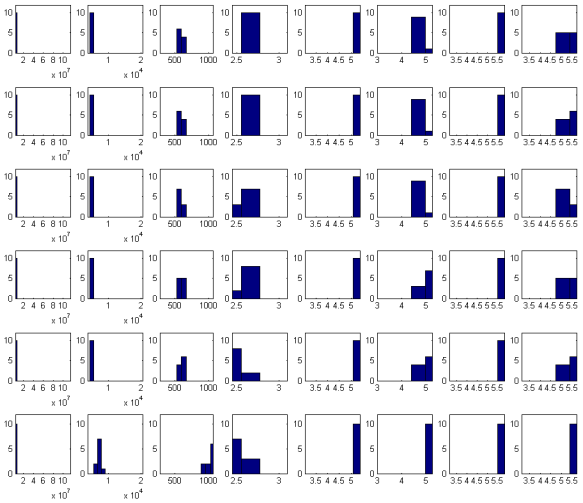


**Figure 7.a Histograms of ordered feature vectors for a ten (10) stamp sliding window** (BC0-NS-AL-M1-AM-MDin)

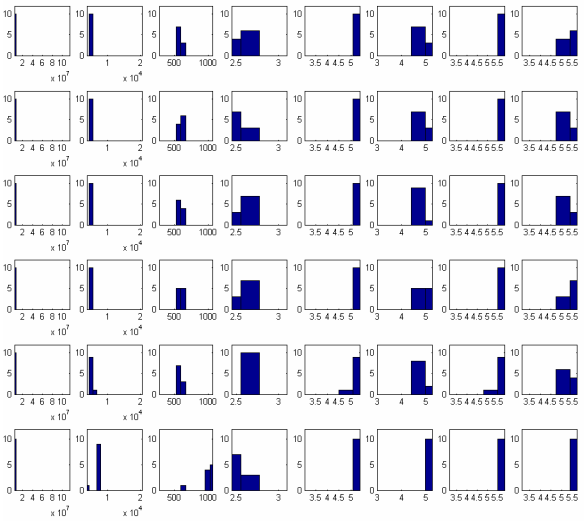


**Figure 7.b Histograms of ordered feature vectors for a ten (10) stamp sliding window** (BC0-NS-AL-M1-AM-MDout)
(from left to right # of packets, # of bytes, # of active flows, entropy of source IP addresses, entropy of destination IP addresses distribution, entropy of source ports, entropy of destination ports, entropy of packet sizes)

The minimum distance from a node of the SOFM after training is given in **Fig. 9** for all time stamps (*t* ranges from 1 to 590). The multiple histogram set includes five more samples corresponding to normal conditions. The sum of all elements of the *Cross-Order Distance Matrix* after thresholding is used obtain the illustrate result, i.e. if

$$abs\left(d(x^S(i), x^{S'}(j)) - d(x^{S'}(i), x^{S'}(j))\right) \le d(x^{S'}(i), x^{S'}(j))$$

and

$$abs\left(d(x^S(i), x^{S'}(j)) - d(x^S(i), x^S(j))\right) \le d(x^S(i), x^S(j))$$

the corresponding element of the *Cross-Order Distance Matrix* is set to zero. *True Positive Rate* vs *False Positive Rate* is evaluated by taking different thresholds and assuming that a value higher than the threshold indicates an anomaly (see **Fig. 10**). A SOFM featuring more nodes yields better results.

**Table 2. Feature histograms for a sliding time window**

| | Centers of histogram bins |
|---|---|
| Number of packets (x10,000) | 404 984 1,563 2,142 2,722 3,301 3,880 4,460 5,039 5,618 6,198 6,777 7,356 7,936 8,515 9,094 9,674 10,253 10,832 11,412 (20 bins) |
| Number of bytes | 3,787 4,975 6,162 7,350 8,538 9,726 10,914 12,102 13,289 14,477 15,665 16,853 18,040 19,228 20,416 (15 bins) |
| Number of active flows in time stamp | 283.5 354.4 425.3 496.2 567.1 638 709 779.9 850.8 921.7 992.6 1,063.5 (12 bins) |
| Entropy of source IP address distribution | 2.4423 2.6608 2.8792 3.0978 (4 bins) |
| Entropy of destination IP address distribution | 3.0266 3.6158 4.2050 4.7942 5.3834 (5 bins) |
| Entropy of source port distribution | 2.9906 3.5558 4.1210 4.6862 5.2514 (5 bins) |
| Entropy of destination port distribution | 3.1532 3.8196 4.4860 5.1524 5.8188 (5 bins) |
| Entropy of packet size distribution | 3.1231 3.7693 4.4155 5.0617 5.7079 (5 bins) |

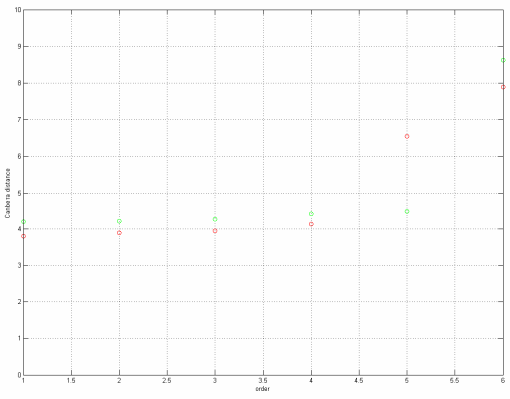

**Figure 8.a Distances vs rank of elements – 6th rank corresponds to anomaly**
(for the histogram feature vectors depicted in Fig. 7.a and 7.b)

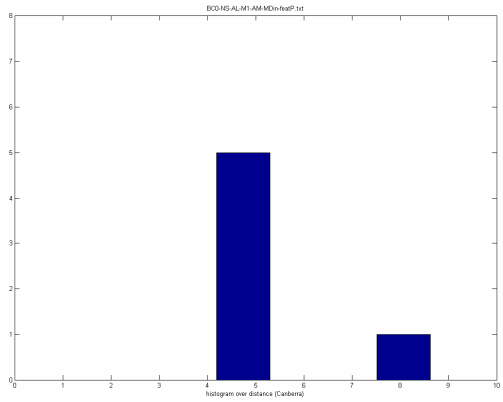

**Figure 8.b Distribution of histogram feature vectors** (depicted in Fig. 7.a) **over ordering distance**

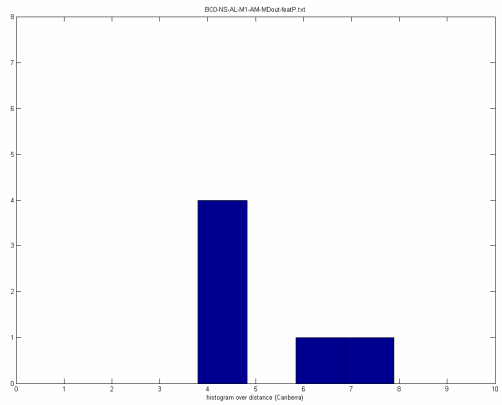

**Figure 8.c Distribution of histogram feature vectors** (depicted in Fig. 7.b) **over ordering distance**

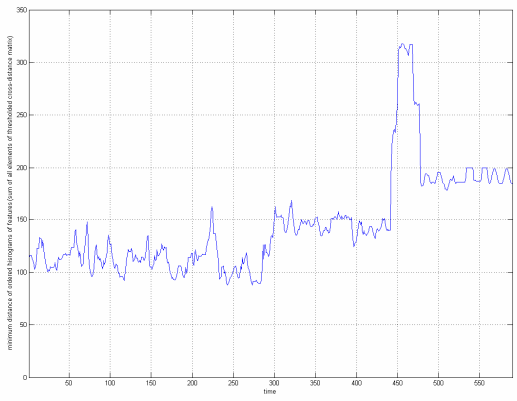

**Figure 9. Outlier distance vs time stamp**
(an attack is introduced during Virtual Machine migration after $t$=300)

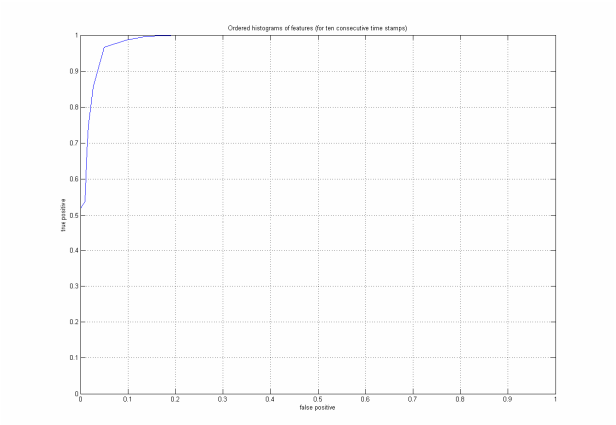


**Figure 10. True Positive Rate (TPR) vs False Positive Rate (FPR)** (for threshold values ranging from 90 to 190 in Fig. 9)

## 5. CONCLUSION

A novel approach for detecting anomalies during typical cloud operation is proposed. The method is based upon ordering histogram feature vectors from several processing sites of the network and training a SOFM. Anomalies are classified according to their type and intensity. An anomaly is detected should the minimum distance from some node of the SOFM exceed a specified threshold. Estimation of the distances between the nodes of the SOFM and the ordered set that contains a feature vector with anomaly measurements is carried out according to a "*method*" applied upon the so-called *Cross-Order Distance* Matrix. One has to specify a certain distance measure, like the Canberra distance which is used in the context of this work in order to estimate the similarity of histograms of feature values within a sliding time window. The proposed approach yields the server-under-attack and the existence of a network anomaly at the same time. Further investigation regarding the convergence of the technique using several "methods" during training is under way.

## 6. ACKNOWLEDGMENTS

The proposed work is inspired by the EU FP7 project SECCRIT (Secure Cloud computing for Critical infrastructure IT) funded by grand agreement no. 312758.

## 7. REFERENCES

[1] Adamova, K., Schatzmann, D., Plattner, B. and Smith. P. 2004. Network Anomaly Detection in the Cloud: The Challenges of Virtual Service Migration. In IEEE *International Conference on Communications* (Sydney, NSW, 10-14 June 2014). ICC, 3770-3775. DOI= 10.1109/ICC.2014.6883908

[2] Tomar K., Tyagi S.S and Agrawal R. 2014. Overview - Snort Intrusion Detection System in Cloud Environment. *International Journal of Information and Computation Technology* 4(3) (2014), 329-334. ISSN 0974-2239.

[3] Simpson, S., Shirazi, N, Hutchison, D., Backhaus, H. 2013. *Deliverable: 4.1 Anomaly Detection Techniques for Cloud Computing*. The SEcure Cloud computing for CRitical infrastructure IT (2013). https://www.seccrit.eu/.

[4] Lakhina, A., Crovella, M. and Diot, C. 2005. Mining anomalies using traffic feature distributions. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures and Protocols for Computer Communications*, SIGCOMM '05. ACM, New York, NY, USA, 217-228.

[5] Pascoal, C., Oliveira, R., Valadas, R., Filzmoser, P. F., Salvador P. and Pacheco A. 2012. Robust feature selection and robust PCA for internet traffic anomaly detection. In IEEE INFOCOM, vol. 2012 (March 2012), 1755-1763.

[6] Wu, N. and Zhang, J. 2006. Factor-analysis based anomaly detection and clustering. *Decision Support Systems*, 42(1) (2006) 375-389.

[7] Zhang, H. 2004. The optimality of naïve bayes. AA, 1(2):3 (2004)

[8] Bishop, C. M. and Nasrabadi, N. M. 2006. *Pattern recognition and machine learning*, vol. 1, Springer, New York.

[9] Rhodes, B. C., Mahaffey, J. A. and Cannady, J. D. 2000. Multiple Self-Organizing Maps for Intrusion Detection. In *Proceedings of the 23rd National Information Systems Security Conference* (2000).

[10] Alharby A. and Imai, H. (2005). Hybrid intrusion detection model based on ordered sequences. *MMM-ACNS 2005, LNCS 3685*, Gorodetsky, V., Kotenko, I. and Skormin, V. (Eds.) (2005) Springer-Verlag, Berlin-Heidelberg, 352-365, ISBN 3-540-29113-X.

[11] Löf, A. and Nelson, R. 2010. Comparing anomaly detection methods in computer networks. In *Proceedings of the Fifth International Conference on Internet Monitoring and Protection* (Washington DC, USA, 2010), ICIMP'10, IEEE Computer Society, 7-10.

[12] Zhang, W. and Yang, O. 2009. A survey of anomaly detection methods in networks. *In International Symposium on Computer Network and Multimedia Technology* (Wuhan, 2009), CNMT 2009, IEEE Computer Society, 1-3.

[13] Patcha A. and Park, J.-M. 2007. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, 51(12) (2007) 3448-3470.

[14] Hudic, A., Hecht, T., Tauber, M., Mauthe, A. and Caceres-Elvira, S. 2014. Towards Continuous Cloud Service Assurance for Critical Infrastructure IT. *International Conference on Future Internet of Things and Cloud* (FiCloud), IEEE, 175-182, DOI 10.1109/FiCloud.2014.36

[15] Barnett, V. (1976). The Ordering of Multivariate Data. *Journal of the Royal Statistical Society Series A*, vol. 139, no. 3 (1976) 318–355.

[16] Rieck, K., Laskov, P. and Muller, K.-R 2006.. Efficient Algorithms for Similarity Measures over Sequential Data: A Look Beyond Kernels. *DAGM 2006, LNCS 4174*, Franke et al. (Eds.), Springer-Verlag Berlin Heidelberg (2006) 374–383.

[17] Cha, S.-H. 2007. Comprehensive Survey on Distance/ Similarity Measures between probability Density Functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, issue 4, vol. 1 (2007).

[18] Stephanakis, I. M., Anastassopoulos, G. and Iliadis, L. 2013. A self-organizing feature map (SOFM) model based on aggregate-ordering of local color vectors according to block similarity measures. *Neurocomputing*, 107 (2013), 97-107 *http://dx.doi.org/10.1016/j.neucom.2012.09.01*