# Web User Identification with Fuzzy Fingerprints

Nuno Homem
TULisbon – Instituto Superior Técnico
INESC-ID
R. Alves Redol 9, 1000-029 Lisboa
nuno.homem@hotmail.com

Joao Paulo Carvalho
TULisbon – Instituto Superior Técnico
INESC-ID
R. Alves Redol 9, 1000-029 Lisboa
Portugal
joao.carvalho@inesc-id.pt

***Abstract*— Fingerprint identification is a well-known technique in forensic sciences. The basic idea of identifying a subject based on a set of features left by the subject actions or behavior can be applied to other domains. Identifying a web user based on a user fingerprint is one such application. This paper considers the problem of extracting fingerprints from web usage logs and matching them with those obtained from a set of known users. It presents an innovative fuzzy fingerprint algorithm based on vector valued fuzzy sets. Used sites are used as base features to create the fingerprint. The assumption is that sites accessed by each user remain approximately stable and are distinctive. The paper presents the proposed algorithm and shows some experimental results that validate the approach. The use of fast and compact algorithms is critical due to the possible huge number of users, and allows this method to be used on near real time.**

***Keywords-component; fuzzy fingerprints, vector valued fuzzy sets, similarity, frequent elements, approximate algorithms, data streams***

## I. INTRODUCTION[1]

In this paper, one considers the problem of extracting a fingerprint from web usage logs and then using that fingerprint to identify the user behind a distinct web session.

Identifying users based in their previously known usage is a known procedure in several areas such as telecommunications and financial fraud detection. For security and judicial purposes, identifying a user based on a specific usage behavior is also relevant. These detection methods are based on the fact that although a suspect or fraudster may present different identification credentials or simply no identification at all, its behavior and its relationships remain the same. Every individual keeps some relationships and habits, either personal or businesswise, stable for some time. Even if some of those relationships and habits change along time (the individual starts working in a different company, a distinct business, marries another person, etc.), some will remain stable (same family, some job, same interests). This provides the basis for the fingerprint as relationships and habits are translated into usage events. By using the destination number as a proxy for the individual behind a specific communication event, we can gather information out of a person's calls to its relationship network. The sites the user accesses in the web are also a clue to the user habits and interests and therefore can be used to identify him in a pool of suspected web sessions.

Within the scope of this paper, one will consider a web session as the sequence of web log records that belong to a single user, either known or unknown. For anonymous web access, one will consider that some method is available to group these records into a single session. Although out of the scope of this paper, user identity could be linked to a single fixed IP address or a single provider login.

Each web log record represents a user visit to a single web site, regardless of the used protocol. The set of known users for whom a fingerprint is available will constitute the suspects for unknown user sessions identification.

Fingerprint identification is a well-known technique in forensic sciences and widely documented. In computer sciences a fingerprint is a procedure that maps an arbitrarily large data item (such as a computer file, or author set of texts) to a much compact information block, its fingerprint, that uniquely identifies the original data for all practical purposes, just as human fingerprints uniquely identify people for practical purposes.

In computer sciences, fingerprints are typically used to avoid the comparison and transmission of bulky data. For example, a web browser or proxy server can efficiently check if a remote file has been modified simply by fetching its fingerprint and comparing it with the fingerprint of the previously fetched copy. Fingerprints are a fast and compact way to identify items.

To serve the user identification purposes, a fingerprint must be able to capture the identity of that user. In other words, the probability of a collision, i.e., two users yielding the same fingerprint, must be small. The fingerprint has also to be robust; a user should be identified even if he changes some aspects of the web use. The idea of identifying users based on a fingerprint is a very appealing one, because identification can theoretically be made on near real time.

To be useful, the fingerprint should comply with some basic criteria:

- Include a minimal set of features that describe the user (the suspect) in a compact format.

[1] This work was in part supported by FCT (INESC-ID multi annual funding) through the PIDDAC Program funds.

- Allow for update operations whenever new information (new logs) on the user is available.
- Allow for a fast comparison process once a new session owner needs to be identified.
- Scalability, i.e., performance should not degrade significantly when the number of sessions and suspects in the pool increases.
- Flexibility, i.e., should allow new suspects to be included in the process, whenever information is available.

This paper proposes a new method for identifying web users given a set of possible suspects. By using the web site access frequencies as a proxy for the individual behind a specific session, one can gather information on the user and identify other unidentified sessions.

The first step in the proposed method is to gather the top-*k* site access frequencies in all known logs of each known user. An approximated algorithm is used for this purpose since classical exact top-k algorithms are inefficient and require the full list of distinct elements to be kept. The Filtered Space-Saving algorithm [1], [7] is used for this purpose since it provides a fast and compact answer to the top-*k* problem although it only gives an approximate solution. This paper defends that the algorithm approximation is not an issue, as a degree of change or randomness has to be expected and incorporated into the detection method.

Once the top-k site access frequencies are available, the fingerprint is constructed by applying a fuzzifying function to each frequency. This paper proposes the innovative method of fuzzifying the set of features based on their order on the top-*k* list instead of their frequency value.

The last part in the process is to perform the same calculations for the session being identified and then to compare this fuzzy fingerprint with all the available users fuzzy fingerprints. The most similar fingerprint is chosen and the session is assigned to the fingerprint author.

## II. Relation with previous work

### A. Signatures and fingerprints

Boltan and Hand [1] and Phua and al. [12] survey multiple methods for fraud detection using statistical and data-mining techniques but no mention is made of research aiming to identify individual fraudsters based on their behavior. A more common, although distinct, approach is the use of usage signatures but to detect behavior changes as in [3], [4] and [5]. Signatures are used to monitor the individual activity and detect changes in that activity, or to compare against known fraud signatures to detect similarity of patterns, not the individuals behind the fraud. In [3] and [5] signatures are created extracting relevant features out of the individual usage, such as number of local, national and international calls, etc., based on pre-defined types of traffic classifications and summarized. In [5] a list of the top-k countries and destinations called, and a count of calls that do not go to any of the top-k is used but with the purpose of detecting changes in the usage.

A recent work by Cormode et al. [1] presents the concept of signature algorithms applied to iterations between individuals and provides some signature schemes to network traffic and telecommunication calls. A generic approach and a theoretical framework for signatures communication graphs analysis are provided. The "signatures" concept has some common points to the proposed user fingerprint. However, one prefers to use the "fingerprint" designation, as the algorithm aims at extracting information about the user that he has not provided knowingly, while a "signature" usually refers to information that was created specifically to identify someone or something. In [1] the feature extraction is exact; the use of approximated algorithms is suggested and several distinct distances are proposed.

### B. Fuzzy Signatures

The fuzzy fingerprint concept is a generalization of the Vector Valued Fuzzy Sets (VVFS) concept introduced by Kóczy [8]. The qualitative meaning of an object is represented by the quantities of the VVFS.

The vector valued fuzzy sets concept has also been used in [9] to introduce the fuzzy signature concept. Fuzzy signatures can model sparse and hierarchically correlated data with the help of hierarchically structured VVFS and a set of not-necessarily homogenous and hierarchically organized aggregation functions.

## III. The Filtered Space-Saving Algorithm

To allow the use of user identification techniques in near real time and for a large number of potential suspects and sessions, a key issue is to be able to extract the relevant features using an efficient algorithm with reduced memory usage. In this case, features are the most frequent visited sites in the user usage. The choice was to use an approximate top-*k* algorithm capable of generating good quality estimates using a reduced memory footprint. The Filtered Space-Saving algorithm [7] was chosen. Filtered Space-Saving, originally presented in [6], is an evolution from Space-Saving algorithm presented by Metwally and al. [10].

The Filtered Space-Saving (FSS) algorithm uses a bitmap counter with $h$ cells, each containing two values, $\alpha_i$ and $c_i$, standing for the error and the number of monitored elements in cell $i$. The hash function needs to be able to transform the input values (site) into an uniformly distributed integer range. The hashed value *hash(x)* is then used to increment the corresponding counter. Initially all values of $\alpha_i$ and $c_i$ are set to 0.

The second storage element is a list of monitored elements $A$ with size $m$. The list is initially empty. Each element contains three parts; the value itself $v_j$, the estimate count $f_j$ and the associated error $e_j$.

The minimum required value to be included in the monitored list is always the minimum of the estimate counts,

$\mu = min\ \{f_j\}$. While the list has free elements, the minimum is set to 0.

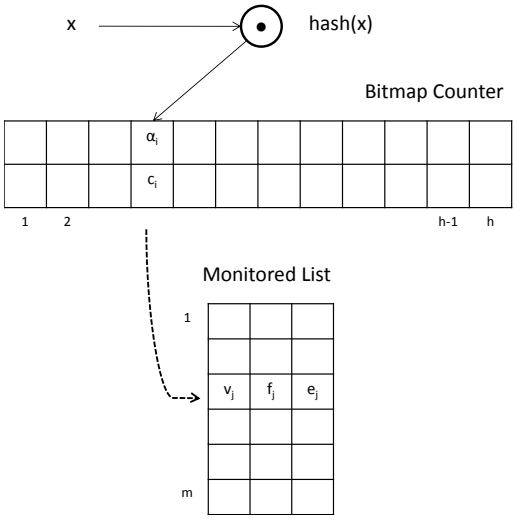


Figure 1 – FSS Algorithm Diagram.

The algorithm is quite simple. When a new element is received, its hash is calculated and the bitmap counter is checked. If there are already monitored elements with that same hash $(c_i > 0)$ the list is searched to see if this particular element is already there. If the element is in the list then the estimate count $f_j$ is incremented. If the element is not in the list then it is checked to see if it should be added.

A new element will be inserted into the list if $\alpha_i + 1 \geq \mu$. If the element is not monitored then $\alpha_i$ is incremented. In fact this $\alpha_i$ stands for the number of elements with hash value i that have not been counted in the monitored list; it is the maximum number of times an element that is not in the list and that has this hash value could have been observed.

```
Algorithm: FSS(h cells, m counters, S stream)

begin
for each element, x, with value w, in S {
    set min to min {fj}
    let i be the hash(x) mod h
    if ci is not 0 {
        if x is monitored {
            let j be the index of x in the list
            increment fj
            continue for next x
        }
    } // this will only be executed if x is not monitored

    if  αi + 1 >= min {
        if list size equals m {
            let m be the index with lower fj
            and for same fj with higher ej
            let k be the hash(x) mod h
            decrement ck
            set αk = fi
            remove vm
        }
        include x in the list in index j
        set vj to x
        set ej to αi and fj to αi+1
        increment counter ci
    } else {
        increment αi
    }
}// end for
end
```

Figure 2 – The FSS Algorithm

## IV. The Fuzzy Fingerprint Algorithm

The main concept behind this algorithm is that users have a stable enough behavior that allows a set of features to be extracted, fuzzified and then compared. The most frequent visited sites of a single user present the required stability.

User web traffic can then be analyzed as a set of site accesses generated by a stable distribution that depends only on the user. Only site visit counts are in fact relevant and used as variables.

The set of sites to consider in the fingerprint should be large enough to allow a comprehensive sample of the user behavior along time. The FSS algorithm requires two parameters, the size of the monitored list $m$ and the size of the bitmap counter $h$. In all the tests the parameters were set proportional to the number of visited sites used in the fingerprint: $m = 3k$, $h = 9k$.

### A. Fuzzy Fingerprint Creation

The full set of known texts are processed through the modified FSS algorithm to compute the approximated top-$k$ list and frequencies for each user. Consider $T_j$ is the set of logs by the user $j$. The result consists of a list of $k$ tuplets $\{v_i, n_i\}$ where $v_i$ is the $i$-th most frequent visited site and $n_i$ the corresponding count estimate.

To create the actual fingerprint, the top-$k$ list has to be fuzzified. The choice of the fuzzifying function is critical and the chosen approach is to assign a membership value to each site in the set based only on the order in the list. In fact, experiments have shown that the order of the frequency seems more relevant than its actual value. The more frequent sites will have a higher membership value.

Several alternative membership attribution functions for each element $i$ of the top-$k$ list are tested in this paper. The simplest one is:

$$\mu_{flat}(i) = \frac{k-i}{k}. \tag{1}$$

Function $\mu_{par}$, inspired in the Pareto rule, where 80% of the membership value is assigned to first 20% elements in the ranking:

$$\mu_{par}(i) = \begin{cases} 1-(1-b)\dfrac{i}{k} & if\ i < a \\ a\left(1-\dfrac{i-a}{k-a}\right) & if\ i \geq a \end{cases} \tag{2}$$

The third function is $\mu_{erfc}$, based on the complementary error function:

$$\mu_{erfc}(i) = 1 - erf\left(\frac{2i}{k}\right), \tag{3}$$

where *erf()* is the Gauss error function. Figure 3 presents the used functions.

The fingerprint based on the top-*k* list (size-*k* fingerprint, $\Phi$), consists on a size-*k* fuzzy vector where each position *i* contains a element $v_i$ and a value $\mu_i$ representing the fuzzified value of $v_i$'s rank (the membership of the rank).

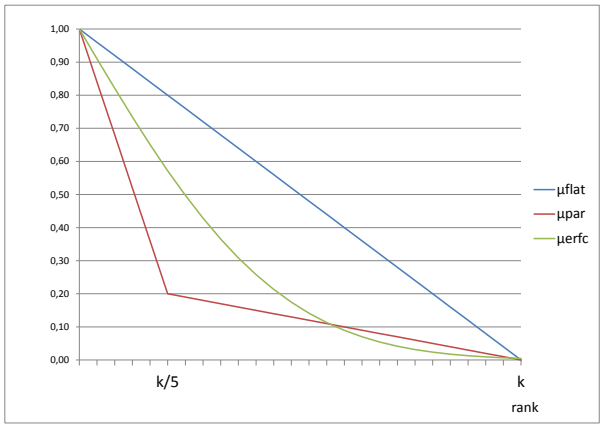


Figure 3 – Fuzzyfying functions

An user *j* will be represented by its fingerprint $\Phi_j = \Phi(T_j)$. Formally, fingerprint $\Phi_j = \{(v_{ji}, \mu_{ji}) | i = 1..k_j\}$ has length $k_j$, let $S_j = \{v_{ji} | i = 1..k_j\}$ be the set of *v*'s in $\Phi_j$.The set of all user fingerprints will constitute the fingerprint library.

### B. *Fuzzy Fingerprint Detection*

In order to find the user of an unknown session *L*, one starts by computing the size-*k* fingerprint of *L*, $\Phi_L$. Then one compares the fingerprint of *L* with the fingerprints $\Phi_j$ of all users present in the fingerprint library. The unknown user is identified as user *j* if he has the most similar fingerprint to $\Phi_L$. Fingerprint comparision, $sim(\Phi_L, \Phi_j)$, is calculated using (4):

$$sim(\Phi_L, \Phi_j) = \sum_{v \in S_L \cup S_j} \frac{\min\left(\mu_v(\Phi_L), \mu_v(\Phi_j)\right)}{k}, \qquad (4)$$

where $\mu_v(\Phi_x)$ is the membership value associated with the rank of element *v* in fingerprint *x*. This function is based on the minimum or Gödel t-norm, but other t-norms could be used.

## V. Experimental Results

### A. *Experimental data set*

Experiments were performed using a data set consisting of 10 days of web logs collected from a large company (including a weekend, with much less records). The data is annonymized but every user in the web access record is identified by a unique tag. Records detailing access to .jpg, .bmp, .gif objects were filtered and are not present in the data set. Filtering the images from the web logs reduces the number records to circa 10%. Note that this company has implemented a web access policy that limits access to some sites

To create the suspect library the logs from the 8 initial days were used, constituting the training set. The sessions to be used as the test set were created with the remaining 2 days of web logs. A total of 21 669 597 records were included in the training set, and 6 621 078 records in the test set. Although both the training and test sets were created using a much reduced sample period, this may in fact be similar to real life situations where data is usually much less than ideal.

A total of 1630 users were considered for these tests, each with a minimum 750 records in the training set and 250 records in the test set. Note that this is a small number of access records; many of the users generate more than a thousand records per day (excluding images).

The site name and port part of the url were used as base features. Site name and port were prefered to the IP address because, first, many sites use multiple servers and IP addresses for load balancing, second, many sites share the same IP address as they are hosted in a single server, third, because with the emergence of cloud hosting services IP addresses are easily changed and the site name is what effectivly distinguishes one site from another. The use of the site name requires also requires some care as some services may use the name to convey some information for user or load distribution, in this case some cleaning is required to transform the site name and port to the generical format of *server.domain:port* or *server.domain*.

Although the sample period is not long it is interesting to see the site access distribution per user. Figure 4 presents the percentage of accesses for each of the most visited sites of each user. On average 90% of the users accesses are made to just 55 sites. Figure 4 also presents distribution of consecutive site accesses, and the mixture of the two distributions. Consecutive site accesses are relevant because one user may visit one site and then visit another. This path may be distinct from other users paths and helps distinguising the users.

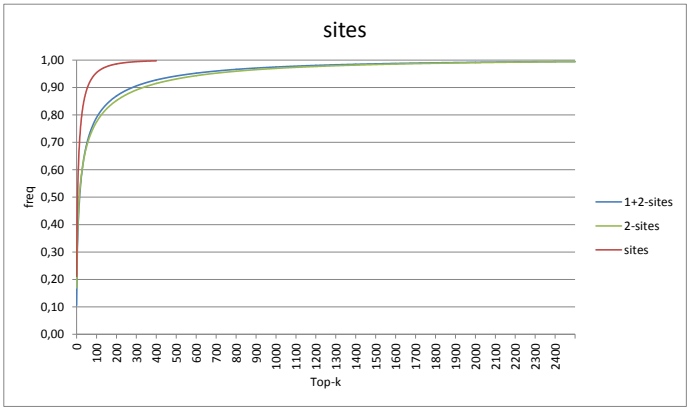


Figure 4 – Web site accesses distribution

### B. *Test methodology*

The distinct fuzzifying functions were tested on each of the available users and sessions. Each test returns a ranked list of the suspects. A test is considered to be positive if the real user is returned in the first position of the ranked list. Accuracy measures the ratio of positive tests for each of the function. For

each of the tested algorithms tests were performed for multiple values of $k$.

To evaluate the accuracy of the proposed algorithms against crisp signature algorithms one will present experimental results obtained using equivalent methods proposed in [2]. In [2] a variety of distance functions were presented to compare signatures. They are generalized from known measures, and take into account both set overlap as well as weighted occurrence. In [2] signatures are $k$ sized tuples with element $v_i$ and a value $w_i$ representing the frequency of the element. Each signature $\Omega_i = \{(v_{ij}, w_{ij}) | j = 1..k_i\}$ has length $k_i$, let $S_i = \{v_{ij} | j = 1..k_i\}$ be the set of $v$'s in $\Omega_i$. The distances considered were adapted from [2]:

$$D_{jac}(\Omega_1, \Omega_2) = 1 - \frac{S_1 \cap S_2}{S_1 \cup S_2} \quad (5)$$

$$D_{dice}(\Omega_1, \Omega_2) = 1 - \frac{\sum_{v \in S1 \cap S2} \min(w_1, w_2)}{\sum_{v \in S1 \cup S2} \max(w_1, w_2)} \quad (6)$$

$$D_{hel}(\Omega_1, \Omega_2) = 1 - \frac{\sum_{v \in S1 \cap S2} \sqrt{w_1 \cdot w_2}}{\sum_{v \in S1 \cup S2} \max(w_1, w_2)} \quad (7)$$

It is easy to verify that all these distance functions yield values in [0, 1]. $D_{jac}$ is based on Jaccard coefficient, where the node weights are not taken into account; it is minimized when S1 = S2, and it equals 1 when their overlap is empty. $D_{dice}$ is an scaled version of the distance based on Dice criterion, which factors in node weights; it gives an added premium if the individual weights in $S_1$ and $S_2$ are similar. $D_{hel}$ is based on Hellinger distance .

To better characterize the classifier response ROC curves are presented for every algorithm. ROC curves are a standard measure in statistics [10]. ROC curves map the ratio of false positives against the ratio of true positives, with the false positives in the x-axis and true positives in the y-axis. ROC curves were constructed by first running every test sample against the suspect library and obtaining a ranked list. Let T be the number of tests and U the number of suspects. The total number of elements in the answer lists is $TU$. The total number of positive tests is $T$ (there is always one correct user for each unknown session) and the number of incorrect tests is $T(U-1)$. The ROC curve starts at the origin (0, 0). For each rank, in ascending order, the number of positive tests $p$ with that rank are used to draw a vertical line upwards by $p/T$. For the same rank a horizontal line is draw to the right by $(T-p)/(T(U-1))$.

Also a standard measure in statistics, the Area Under the ROC Curve (AUC) [10] can also be computed. AUC is an indicator of the overall accuracy of the classifier, if the AUC is 0.5, the classifier is no better than random selection; higher values indicate better precision. An AUC of 1 indicates a perfect classifier.

### C. *Single site fuzzy fingerprints*

The first set of experiments use single site name and port fingerprints. Figure 5 presents the accuracy of fuzzy fingerprints using the three fuzzifying functions described and the results obtained using the three crisp distance functions. The difference between fuzzy and crisp algorithms is very significant. The fuzzy fingerprints deal better with the very sparse nature of the web accesses and significant differences of use in the training and test periods. Differences between the three proposed fuzzifying functions are not very relevant, the Pareto based function achieves the best results, followed closely by the complementary gaussian error function. These functions weight more the most frequent elements.

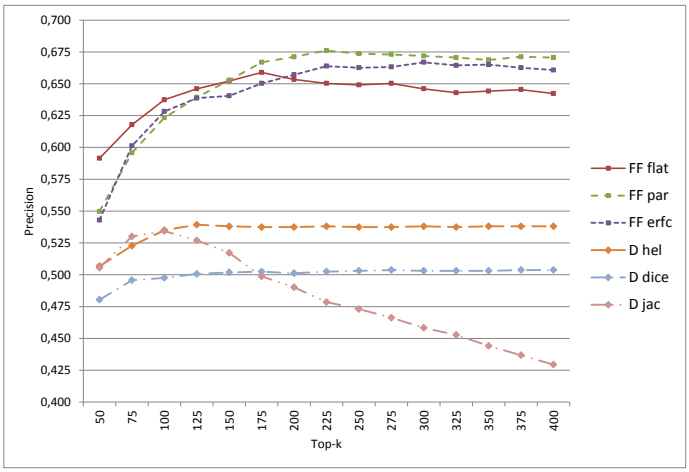


Figure 5 – Algorithms accuracy for single web site access

Figure 6 presents the ROC curve for each of the tested algorithms. To better show the differences Figure 7 presents the zoom of the most relevant area of the ROC.

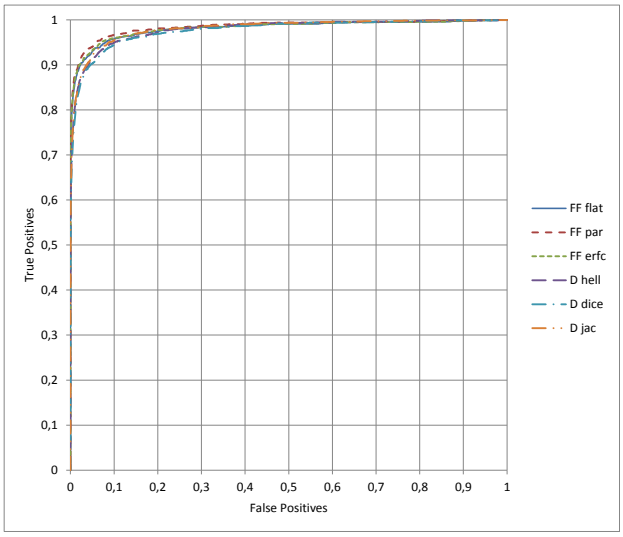


Figure 6 – Algorithms ROC for single web site access

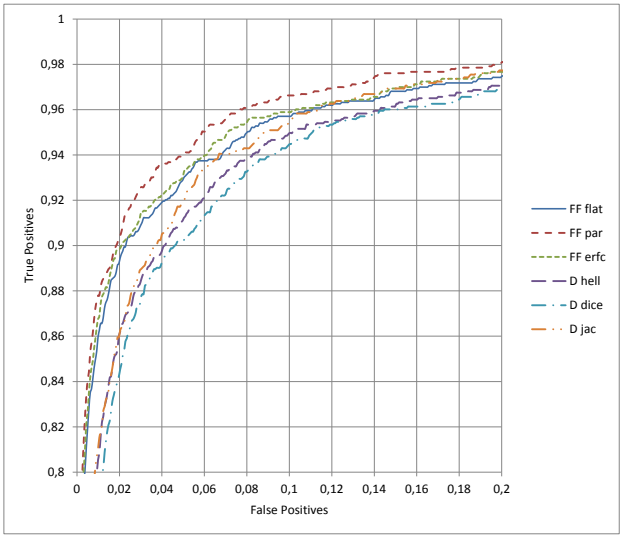


Figure 7 – Detailed ROC for single web site access

### D. *Consecutive sites fuzzy fingerprints*

The second set of experiments uses pairs of consecutive site name and port fingerprints. Figure 8 presents accuracy for each of the algorithms. Both crisp and fuzzy algorithms present better results, both can use the extra information contained in the order of the accesses. But this performance increase is only achieved at the expense of much higher number of top-*k* elements. Fuzzy algorithms perform much better than crisp algorithms but is interesting to see that the worst performing algorithm in the single site tests, the Jaccard based distance performes much better with consecutive sites. Fuzzy fingerprints using the Pareto based function achieves the best results, followed closely by the complementary gaussian error function.

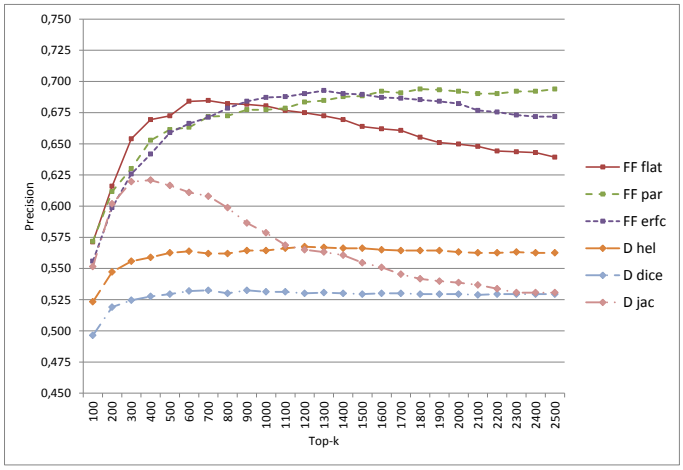


Figure 8 – Algorithms accuracy for consecutive web site accesses

Figure 9 presents the ROC curves. To better show the differences Figure 10 presents the zoom of the most relevant area of the ROC.

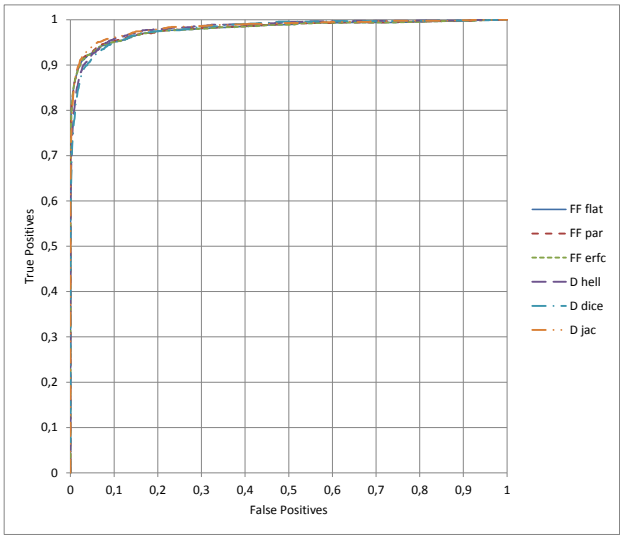


Figure 9 – Algorithms ROC for consecutive web site accesses

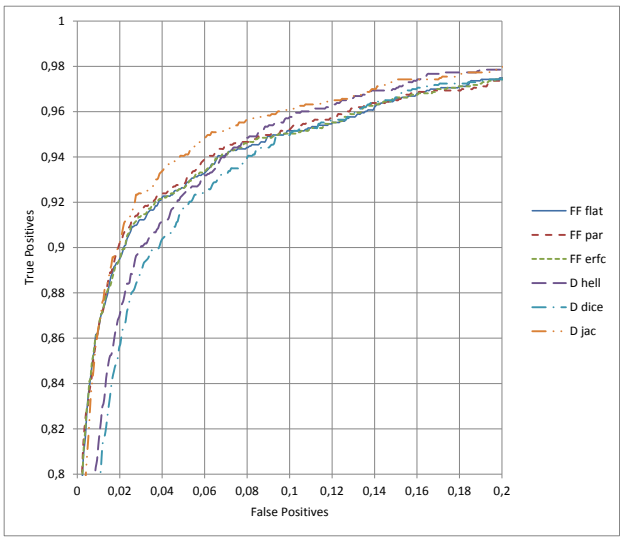


Figure 10 – Detailed ROC for consecutive web site accesses

Interestingly the ROC shows that the Jaccard based distance behaves quite well overall. Although the accuracy of the top choice is not the best of all algorithms, it generates less false positives when the list is enlarged.

### E. *Mixture fuzzy fingerprints*

The final set of experiments use both the single site and pairs of consecutive site name and port fingerprints. Both elements are mixed in a single top-k list. Figure 11 presents the accuracy for each of the algorithms.

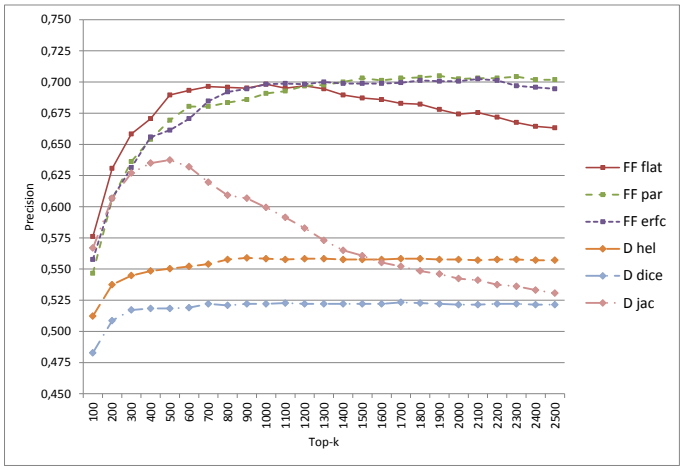


Figure 11 – Algorithms accuracy for mixture fingerprints

Figure 12 and Figure 13 present the corresponting ROC curves.

There is a slight increase in performance of all fuzzy algorithms and for the Jaccard based distance, a slight reduction for the Dice and Hellinger based distances. The most relevant benefict for the use of the mixture is that performance remains stable for a wider range of k. For Pareto and complementary gaussian error based functions the performance remains stable between k values of 1000 to 2500.

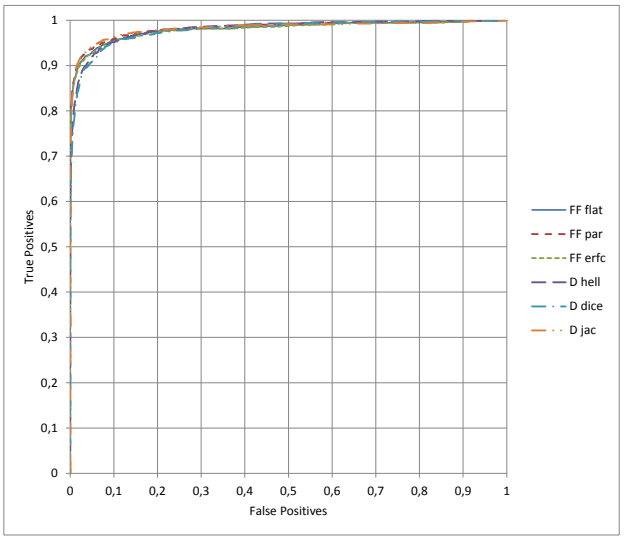


Figure 12 – Algorithms ROC for mixture fingerprints

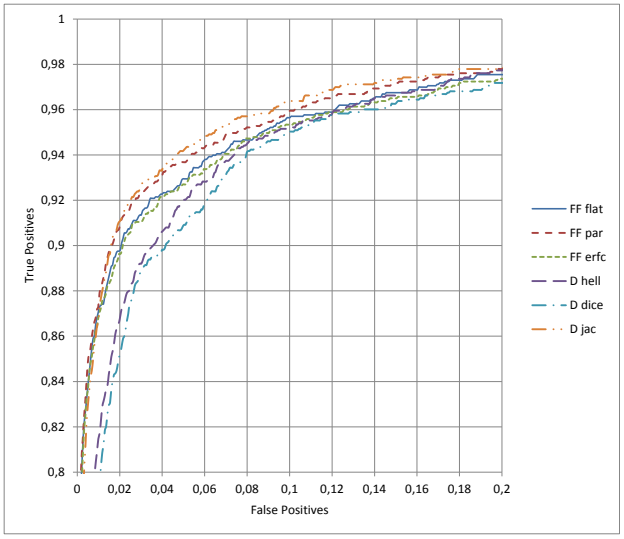


Figure 13 – Detailed ROC for mixture fingerprints

Table I presents the peak performance values for all of the experiments and algorithms. It includes the Area Under the ROC Curve (AUC) [10]. AUC is an indicator of the overall accuracy of the classifier, if the AUC is 0.5, the classifier is no better than random selection; higher values indicate better precision. An AUC of 1 indicates a perfect classifier.

TABLE I. RESULTS FOR EXPERIMENTS WITH WEB SITE ACCESSES

| | | **Fuzzy Fingerprints** | | | **Crisp Distances** | | |
|---|---|---|---|---|---|---|---|
| | | ***flat*** | ***par*** | ***erfc*** | ***hell*** | ***dice*** | ***jac*** |
| **Tests** | | 1630 | 1630 | 1630 | 1630 | 1630 | 1630 |
| **Users** | | 1630 | 1630 | 1630 | 1630 | 1630 | 1630 |
| **Single Site** | ***Peak k*** | 175 | 225 | 300 | 125 | 275 | 100 |
| | ***Peak Accuracy*** | 0,6589 | 0,6761 | 0,6669 | 0,5393 | 0,5037 | 0,5344 |
| | ***AUC*** | 0,9811 | 0,9848 | 0,9819 | 0,9792 | 0,9765 | 0,9810 |
| **2 Sites** | ***Peak k*** | 700 | 1800 | 1300 | 1200 | 700 | 400 |
| | ***Peak Accuracy*** | 0,6847 | 0,6939 | 0,6926 | 0,5675 | 0,5325 | 0,6209 |
| | ***AUC*** | 0,9802 | 0,9809 | 0,9802 | 0,9822 | 0,9800 | 0,9840 |
| **1+2 Sites** | ***Peak k*** | 1000 | 1900 | 2100 | 900 | 1700 | 500 |
| | ***Peak Accuracy*** | 0,6982 | 0,7049 | 0,7025 | 0,5589 | 0,5233 | 0,6374 |
| | ***AUC*** | 0,9807 | 0,9829 | 0,9798 | 0,9809 | 0,9786 | 0,9826 |

## VI. Conclusions and Future Work

This work shows how fuzzy methods may be used to identify the users behind an unidentified web session. The use of simple fuzzy techniques based on approximate algorithms leads to very interesting results.

The fact that the results remain stable for a wide range of values of *k* and for several fuzzification functions shows that the proposed method is robust.

The large number of suspects it supports and the ability to include new suspects or update existing user fuzzy fingerprints is critical to the use of the method in long-term detection processes. New fuzzy fingerprints can be created and added to the suspect's library at any time, and new session logs from known users can be added to the fuzzy fingerprint. Update to one fuzzy fingerprint does not influence all the others.

In this work the method is applied to identifying a author within a list of possible suspects but it can also be applied to the single author problem or to the one or none author within a list by considering the distance (or the inverse comparison) between fuzzy fingerprints as a distinguishing feature. One can then apply a binary classifier, such as a Bayes classifier to determine if the author is a probable one. There is however a significant risk in reducing the complexity of a very high level dimension problem to a single dimension. A candidate should only be tested in this way if all other candidates have been eliminated (i.e., it should only be applied to the top of a list of suspects).

The use of order information, by using consecutive events as base features, enriches the fingerprints, even with such as sparse data as web site names. One can expect that this approach could lead to even better improvements in domains where event order is more relevant.

The proposed method should not be seen as a specific method; it can be used in other domains to identify individual behavior based on events. The proposed method will identify individuals as long as they present a stable event distribution. Future work should also test the inclusion of additional features available in the web logs.

## References


[1] R. Bolton and D. Hand, Statistical Fraud Detection: A Review, Statistical Science, Vol. 17, No. 3, 235–255, 2002

[2] G. Cormode, F. Korn, S. Muthukrishnan, Yihua Wu, On signatures for communication graphs, IEEE 24th International Conference on Data Engineering (ICDE), 2008.

[3] C. Cortes and D. Pregibon, Signature-Based Methods for Data Streams. Data Mining and Knowledge Discovery 5: 167-182, 2001

[4] T. Fawcett, and F. Provost, Activity monitoring: Noticing Interesting Changes in Behavior. Proc. of SIGKDD99, 53-62, 1999

[5] P. Ferreira, R. Alves, O. Belo and L. Cortesão, Establishing Fraud Detection Patterns Based on Signatures, Proceedings of the 6th Industrial Conference on Data Mining, ICDM2006, Lecture Notes in Computer Science, Springer, 2006

[6] N. Homem and J. Carvalho, Estimating Top-k Destinations in Data Streams, Computational Intelligence for Knowledge-Based Systems Design, Springer Berlin / Heidelberg, pages 290-299, 2010.

[7] N. Homem and J. Carvalho, Finding top-k elements in data streams, Information Sciences, 180(24), pp. 4958-4974, Dec. 2010, Elsevier.

[8] L. Kóczy, Vector valued fuzzy sets, BUSEFAL- BULL STUD EXCH FUZZIN APPL, pages 41–57, 1980.

[9] L. Kóczy, T. Vámos, G. Biró, Fuzzy signatures, in: EUROFUSE-SIC99, 1999.

[10] S. Mason and N. Graham, Areas beneath the relative operating characteristics (ROC) and relative operating levels (ROL) curves: Statistical significance and interpretation. Q. J. R. Meteorol. Soc, 30:291–303, 1982.

[11] A. Metwally, D. Agrawal and A. Abbadi, Efficient Computation of Frequent and Top- k Elements in Data Streams, Technical Report 2005-23, University of California, Santa Barbara, September 2005.

[12] C. Phua, V. Lee, K. Smith and R. Gayler, Comprehensive Survey of Data Mining-based Fraud Detection Research, Artificial Intelligence Review, 2005.