Extracting Evidence from Filesystem Activity using Bayesian Networks

Muhammad Naeem Ahmed Khan, Chris R. Chatwin and Rupert C.D. Young
m.n.a.khan@sussex.ac.uk, c.r.chatwin@sussex.ac.uk, r.c.d.young@sussex.ac.uk

Abstract - This research aims to ascertain filesystem access patterns produced by different application programs, and evaluates their potential utility in improving digital forensic analyses. The access patterns produced by the proposed methodology can serve as a decision support system for determining the possible execution of certain applications in the event of computer misuse. For this purpose, we propose the use of a causal Bayesian network that summarizes the most important relationships among integral parameters relating to filesystem activities such as access, creation, modification, file deletion, audit logs, registry entries and the manner in which the applications manipulate these parameters.

Determining the state of a filesystem at a particular period of time is vital for conducting digital forensic analyses. Herein, we describe a Bayesian network-based technique to determine the state of a computer filesystem in terms of the program execution and files manipulated during some specific time period. Specifically, we discuss the construction of a Bayesian network from our prior knowledge of the manipulation of the filesystem and metadata information by a set of applications. The variations among the execution patterns of different applications indicate that the Bayesian network-based model is an appropriate tool, due to its ability to enable pattern learning and detection, even from an incomplete dataset. The focus of this paper is to highlight the merits of the Bayesian methods for learning, with regard to the techniques used for supervised learning in ordinary neural networks.

Key words: Digital Forensics, Digital Evidence, Bayesian Learning, Bayesian Belief Networks, File-system Activity Analysis, Event Reconstruction

I. Introduction

The field of computer forensics has developed rapidly over the past decade, due to a sharp rise in computer-related crimes. The use of computer and network technologies has led to new avenues for committing crimes in novel ways. Moreover, technology has made it easier for criminals to hide information about their crimes and related activities. The occurrence of credit card fraud and identity theft by electronic means is on the rise [1]. The fast growth of the internet has contributed to the proliferation of such cyber-crimes as child pornography, gambling, money laundering, financial scams, propaganda and promoting extremism [2]. From a digital forensics perspective,
the construction of the past events that occurred on a digital media serves as a baseline for the digital investigation. In order to reconstruct the crime scene scenario, it is imperative to analyze the traces left on the filesystem and identify the different applications and tools used by digital crime perpetrators.

The key challenge of forensic analysis is to manage and assimilate many large and diverse datasets. Identifying evidence from large data volumes is tantamount to finding a needle in a haystack [3]. Toward solving this dilemma, the use of statistical techniques may enable one to sort through the evidence in the appropriate set of data. Bayesian belief networks and Bayesian decision theory are models that have been used effectively for a number of years. While these models incorporate prior knowledge extracted from selected sample datasets, Bayesian techniques have also been employed successfully with incomplete datasets [4]. This feature is intuitively appealing since many datasets that could be useful in tracking forensic cyber crimes are usually not complete.

Primary sources of evidence used in digital investigations include audit logs and filesystem attributes [5]. Preparing a timeline of the filesystem activity on a computing system is crucial for conducting the digital forensic investigation [1]. Machine-learning techniques, which combine data mining and statistical methods, are widely used in engineering and computer science disciplines because they can learn efficiently from large volumes of data in a short span of time [6]. Thus, a machine learning–based event reconstruction methodology could serve an important role in digital forensic analyses [7]. A crucial consideration for generating a timeline is to describe the source application program that was used to manipulate a file. Although some of the logs, including Windows “Security” logs, keep track of this information to some extent, the filesystem data lacks this information and only accounts for the MAC–times and file organization on the storage media. Therefore, in the absence of logs data or log–scrubbing, the same feature is required to be extracted from the filesystem activities data.

This paper describes a Bayesian network–based technique to determine the possible execution of certain applications, and the files they could have manipulated in the past. We propose a Bayesian networks–based model for extracting and encoding knowledge from the data relating to the filesystem of a computing device such as the New Technology File System (NTFS) metadata information, audit logs and registry keys. The information summarized by the proposed procedure will indicate the creation, modification, accession and deletion of documents and images caused by executing different application programs. This information will facilitate the preparation of a consolidated report on the possible use of different applications for manipulating different files during a computer misuse incident. This methodology would be helpful in understanding specific tasks, such as determining which of the applications were running on the system at that particular time, and which systems and data files were created, accessed, modified or deleted at a particular time.

II. Background and Motivation

Digital evidence plays a vital role in enabling investigators to pursue computer–related crimes. Digital forensic analysis of electronic data storage devices enables one to uncover digital evidence for an investigation. Digital forensics is applied typically after a computer system’s security has been compromised, and to determine the reason behind the system’s abnormal behaviour [5]. A methodology is required to guide the investigator, and allow him/her to infer conclusions from the digital evidence. Carrier [8] has categorised Digital forensics into three major phases: Acquisition, Analysis and Presentation. The Acquisition Phase requires saving the state of a digital system so that it can be analyzed later. The Analysis Phase entails examining the acquired data to identify evidence of malicious activity; this aspect of the method is the focus of our paper. The Presentation Phase summarizes the evidence extracted from the investigation, and draws conclusions for potential use in legal proceedings.
Some of the techniques used in intrusion detection (e.g. anomaly detection) overlap with digital forensics. The earliest proposed methods for intrusion detection focused on the application of statistical methods to identify anomalous activities [9]. More recent anomaly detection methods employ a wide variety of classification schemes to identify unusual activities. These schemes include rule induction, artificial neural networks, fuzzy set theory [10] and classical machine learning algorithms. Ryan et al. [11] proposed that artificial neural networks could be used as alternatives to the statistical analysis component of the anomaly detection system. In contrast to statistical techniques, machine–learning techniques are well–suited to learning patterns without prior knowledge about the data. Repeated Incremental Pruning to Produce Error Reduction (RIPPER) [12] is a well–known rule–based anomaly detection tool that employs machine learning to predict an operating system’s supervisory calls. Generalization capabilities of machine learning methods such as neural networks can be employed in creating user profiles based on selection and subsequent classification of their permissions and system usage habits [13]. Regarding the large amount of data stored in an unstructured format, the commonly used rule–based and data mining methods do not work effectively without an ample amount of preprocessing and restructuring of the data.

Numerous researchers have tried a variety of statistical approaches for facilitating the digital forensic analysis [14, 15]. Among these approaches, Bayesian statistics is the most widely employed since the Bayes’ rule can also be used as a classifier [16]. Bayesian classification is a generative approach to data classification. Brewer et al. [17] proposed a method based on a hidden Markov model–based approach to detecting interactions among multiple suspects by coupling the Markov models. Qin et al. [18] demonstrated how causal networks are effective in drawing probabilistic inferences to correlate and analyze digital attack scenarios to generate security alerts accordingly. One distinct advantage of a Bayesian neural network method over the conventional feedforward backpropagation neural network method is that Bayesian can assess the confidence associated with the network’s predictions [19].

Simple Bayesian classifiers are quite popular, and have been demonstrated to perform especially well in classification tasks involving textual data [20]. The Bayesian probabilistic approaches make strong assumptions about how the data is generated, and hypothesize a probabilistic model that embodies the underlying assumptions. The Bayesian approach uses a collection of labeled training examples to estimate the parameters of the generative model. Bayesian networks offer a robust means of handling the complexity of information and discovering valid patterns in the data [21]. Classification using new examples is performed with Bayes’ rule by selecting the class that is most likely to have generated that example. The task of learning models for many real world problems requires incorporating domain knowledge into learning algorithms in order to enable accurate learning from a realistic volume of training data.

Being motivated by the demands of this challenging field, we present a Bayesian belief network–based model for determining the possible execution of certain applications on the system, and identifying the files that could have been manipulated by these applications. The model will enable us to construct a timeline of the past events that occurred on a system. Our approach is based on collecting the filesystem activities from the disk image, and using the Bayesian network to determine if and when particular applications, accompanied by appropriate likelihood measures, ran upon a computer system.

III. Probabilistic Computational Model

The Bayesian probability of an event is one’s degree of belief in the occurrence of that event. Bayesian probability does not require repeated trials. This is an important distinction between physical and Bayesian probabilities. Normally, the method of measuring a degree of belief for a certain event is commonly known as probability
assessment. Conversely, the Bayesian definition of probability endures a general criticism that such probabilities seem to be somewhat arbitrary [22]. Despite this observation, the assessment probabilities are not susceptible to minor variations in actual probabilities, and thus are useful during the decision-making process. Bayesian approaches are at the core of many artificial intelligence programs and computer software, including the well-known Microsoft Office products. In an article published in the October 28, 1996 issue of the Los Angeles Times, USA, Bill Gates, Chairman of the Microsoft Corporations stated, “Microsoft’s competitive advantage lies in its expertise in Bayesian networks”.

Because of the successful implementations of Bayesian techniques to extract knowledge from data sets [23], we chose to employ these methods for forensic analysis of filesystem data. The idea is to model a Bayesian network that can learn from inherent data based on prior knowledge already extracted from a similar scenario. We first extract filesystem activities from the disk image and pull out the relevant features to store in a database. Joint and conditional probabilities tables were then constructed over the entire dataset for all the manipulated files during specific time periods. These probabilities are then used to construct a Bayesian network to classify the filesystem activities from potential file manipulations noticed because of the execution of certain applications. The trained Bayesian network is then used for evaluating the test dataset. A block diagram of a Bayesian classification based computational model is illustrated in Figure 1.

A. Bayesian Approach to Data Classification

Bayes’ rule is a simple mathematical formula used for predicting an event on the basis of conditional and marginal probabilities. Bayes’ rule incorporates the prior knowledge about the data, which is usually available in the form of a probability density based on the results of other experiments, expert opinion or any other source of relevant information. Suppose we have a dataset $D$ consisting of $N$ observations of different application programs’ filesystem access patterns, and a hypothesis $h(D)$ that contains the correct class information of various application programs from the representative training dataset. Here, our assumption is based on the supposition that there are $K$ distinct classes of application programs (i.e., $h(D) \in \{1, ..., K\}$). Thus, we can conclude that the training instances are drawn from the model $(D, h(D))$. The goal of our data classification experiment is to find $h(D_{new})$ given a new dataset $D_{new}$. If we append the new data instances to the already classified data, the problem can be redefined to determine the
likelihood of $N+1^{th}$ and successive observations satisfying any class of the previously determined patterns. For this purpose, we used the Bayesian rule to calculate the posterior probability of the subsequent observations. For a given dataset $D$ and hypothesis $h$, Bayes’ rule is defined as:

$$p(h|D) = \frac{p(D|h)p(h)}{p(D)}$$

where,

- $p(h)$ = prior belief (i.e., probability of hypothesis $h$ before seeing any new data)
- $p(D|h)$ = likelihood probability (i.e., probability of the data if the hypothesis $h$ is true)
- $p(D)$ = data evidence (also known as marginal probability of data). This is also known as a normalizing factor so that the total probability does not exceed ‘1’.
- $p(h|D)$ = posterior probability (i.e., probability of hypothesis $h$ after having seen the data $D$)

The fundamental nature of the Bayesian theorem is to provide a mathematical tenet to change the existing beliefs in the data on the basis of new evidence. Bayesian theorem signifies how observations alter beliefs, as the prior probability is replaced by the posterior probability after the user obtains additional information from the observed data. Given a chronological dataset of filesystem activity, the potential manipulation of various files, one after another by one or more applications, can be assessed by using the previously calculated conditional probabilities in the subsequent stages. Since Bayes’ rule provides a method of sequential updating of probabilities as additional observations become available, and as more and more filesystem activity data would be used, the continued application of Bayes’ rule would refine the application classification process.

The value of Bayes’ rule can be appreciated by considering an example [24]. Suppose a malignant cancer test has 97% accuracy and reports 3% false positives. A patient tests positive for this test; whereas, the statistical records show that just 1% of the entire population of the town is suffering from this disease where the patient is living. Does the physician, upon seeing the test report, reach the conclusion that the patient is suffering from cancer, or should the patient be retested on the basis of the information about the overall cancer case histories in the town? The Bayes’ rule can help the physician reach a decision on whether or not to repeat the test. The available information can be described in the form of probabilities as:

- $p(T|C) = 0.97$ – likelihood probability of positive test when the patient has cancer
- $p(T|\sim C)=0.03$ – probability of positive test when the patient does not has cancer
- $p(C)=0.01$ – prior probability of the cancer cases for the town’s population
- $p(\sim C)=0.99$ – converse of prior probability

The posterior probability of a patient having cancer when the test result is positive can be calculated using the Bayes’ rule.

Since the $a$ posteriori probability indicates that there is a 24.6% chance of the patient having cancer on the basis of the evidence from the first test, it is clear that a second test is required. The main feature of the Bayesian approach is that the results obtained in the first stage are used as an $a$ priori probability in the second stage. If the second test result is also positive, then the previously obtained $a$ posteriori probability of 0.246 will be used as $a$ priori data to calculate the next $a$ posteriori probability. In the case in which the second test is also positive, then the $a$ posteriori probability would be:

$$p(C|TT) = \frac{0.97x0.246}{0.97x0.246+0.03x0.754} = 0.913$$

The new $a$ posteriori probability after conducting second test shows that it is very likely that the patient has cancer.
B. Construction of a Bayesian Belief Network

The Directed Acyclic Graph (DAG) structure of Bayesian networks contains nodes representing domain variables, and the arcs between the nodes represent probabilistic dependencies. We used nodes to represent filesystem attributes. The process of building a Bayesian network is described below.

Consider that we have statistics of creation, modification, accessing and deletion of N number of files during a time interval t. On the basis of the filenames in association with their file types, we build a directed acyclic graph of their potential access by different application programs. The type of file manipulated during a specific time period can be verified by its extension, in conjunction with its header information (also known as a signature). On the basis of prior knowledge extracted from the probability table, an edge (arrow) is marked from the specific filename to all the potential application programs to which it could be related. All of the files having similar types would be interlinked to minimize the number of edges emanating in the graph.

During Bayesian network construction, we built a DAG that encodes assertions of conditional independence. Since a Bayesian network for a dataset \( X = \{X_1, \ldots, X_n\} \) determines a joint probability distribution for \( X \), we used the Bayesian network to compute any probability of interest. From the chain rule of probability, we calculated the probability of the occurrence of an event as:

\[
p(x) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1})
\]

Once we constructed a Bayesian network from prior knowledge or data, we can determine various probabilities of interest from the model. For example, in our problem of finding if an application was launched or not on a system, we want to know the posterior probability of the applications from the available observations of various system and data files accessed during a specified time period. In general, computing a probability of interest given a model is known as “probabilistic inference”. A practical example of the entire procedure is provided in the following section.

IV. Experimentation Methodology

Due to its widespread usage and the fact that it is most commonly attacked [25], we chose the Windows XP operating system to conduct the trials of our experiments. However, much of our methodology can be tailored to function with other operating systems, such as Unix/Linux or previous versions of Windows/DOS.

We conducted a series of experiments by running different application programs. We developed a program module in the C-Sharp (C#) programming language to capture the access patterns of various files by different applications. This module uses the features of a library class FileSystemWatcher of Microsoft’s .Net (dot Net) technology for filesystem monitoring. The FileSystemWatcher class enables the establishment of a connection to the filesystem through which every specific change, such as creation, modification, deletion and renaming of directories and files of the entire filesystem can be monitored online.

We obtained footprints of the filesystem activities by running different applications, namely MS-Word, Notepad, Power Point, Excel, Endnote, Internet Explorer, Adobe Acrobat Reader, Windows Media Player, RealPlayer and MATLAB. We ran these applications separately and simultaneously, to determine how they accessed various systems and data files. This process was conducted numerous times with different combinations of the application programs to collect sufficient amounts of filesystem activities data for classification purposes. For this incentive, we recorded the time of the launch, run and closing of each individual application program, which helped us to sift through the dataset and associate different filesystem activities to the corresponding application programs. We identified some overlaps, however, as some of the DLL files were shared among multiple
applications, and certain data files can be accessed potentially by multiple applications, such as text and image files. In machine-learning methodology, this procedure is commonly known as “labeling the data”. The ultimate purpose of this approach was to determine the filesystem manipulation pattern of the application programs. Using this generated dataset, we constructed the joint and conditional probabilities of file manipulation by different applications.

Our experimental procedure was conducted as follows. First, we categorized the applications by assigning them with numeric labels (1–10). All of the files that were manipulated during the execution of these application programs were associated to their corresponding application by assigning the same labels as their source application. The files that were accessed by multiple applications received multiple labels. To resolve the issue of overlapping, we calculated joint probabilities based on the frequency at which a file was accessed by each individual application. The filesystem activity information was stored in a database in a structured format incorporating important parameters of filesystem activity, such as the type of access, Modification, Access and Creation (MAC) times, directory path, filename, user’s information, registry key, registry value and classification label. The joint and conditional probabilities were stored separately, and were used later in the construction of the Bayesian network. The records stored in the database were preprocessed appropriately to ensure their suitability for statistical analysis and training on the Bayesian network. Subsequently, all of the data were trained on a Bayesian model of a neural network.

In this work, we employed a Bayesian neural network using a multi-layer perceptron (MLP) model for automatic classification by first training the neural network from a dataset of labeled examples. Theoretically, the Bayesian neural network is a one-layer perceptron network implementing a naive Bayesian classifier, as a naive Bayes’ classifier is based on the assumption that different attributes of the dataset are orthogonal. Hidden layers are introduced in the MLP neural network model when classifying the nonlinear data; therefore, we enhanced the Bayesian neural network by introducing a hidden layer to address the data independence requirements. The Bayesian neural networks handle inputs in the form of probability distributions. Therefore, we used Bayesian statistics to estimate the probabilities from filesystem dataset to set up the initial weights in the network. Subsequently, the Bayesian neural network model was evaluated from the test part of the dataset. The classification results were more promising than an ordinary neural network. Showing the joint probability distribution tables and describing the procedure for calculating the prior probabilities for ten application programs would be too long. Therefore, for the sake of brevity, we described the construction of a Bayesian belief network for three application programs in section 4.2.

A. Event Classification Process

During this process, we decided initially upon the number of classes for the data on the basis of different application programs. We then extracted features from the manipulated files, including their names, extensions, types, sizes and parent directories. The extracted features were trained on a Bayesian classifier to learn the filesystem activity pattern of different classes of application programs. The trained classifier was then used for classifying the new filesystem activities. The block diagram of the overall procedure is shown in Figure 2.

The most commonly used prominent practical
approaches to Bayesian learning for multilayer perceptrons are the evidence framework [26] and Monte Carlo methods [26,27]. The evidence framework involves Gaussian approximation to the posterior probability density. We employed this approach because of its promising results and its universal use in the engineering and computer science data analysis problems involving statistical inferences. For a set of training data and labeled instances \(D = \{(x_i, t_i)\}_{i=1}^{n}\), the multilayer perceptron model in the Bayesian framework is described in terms of the posterior probability density over the weights \(p(w|D)\).

B. A Bayesian Belief Network for Three Application Programs

As a case study, we present an example in which the network must decide which of the three application programs \(A_1, A_2\), and \(A_3\) are considered to have potentially manipulated a file \(F\). The causal effect of this scenario is presented in the form of DAG, as illustrated in Figure 3.

Consider that we have already calculated the probabilities of manipulating a file \(F\) (or type of the affected file) by the application programs \(A_1, A_2\), and \(A_3\). We treat the chances of manipulation of the file by the application programs \(A_1, A_2\), and \(A_3\) as true probability, and denot them as \(p(A_1)\), \(p(A_2)\), and \(p(A_3)\), respectively. Alternatively, the chance of zero manipulation of a file by the application programs \(A_1, A_2\), and \(A_3\) is considered as a false probability, and is denoted as \(p(A_1')\), \(p(A_2')\), and \(p(A_3')\), respectively. The probabilities shown in Table I were extracted from the exemplary dataset sample for exclusive execution of the application programs.

The conditional probabilities for simultaneous execution of the three application programs are given in Table II. Typically, multiple conditional probabilities can be estimated for every possible combination. However, some of these probabilities can be calculated from other combinations, because those for each state should sum to 1. In this particular case for three application programs, the total distinctive possibilities are \(2^3 (=8)\). In the case of ten application programs, the total number of possible unique conditional probabilities would be \(2^{10} (=1024)\), despite the fact that a large number of conditional probabilities would be missing. These probabilities would correspond to the conditional independence among multiple applications, due to their distinct filesystem manipulation patterns.

On the basis of known joint probabilities provided in Table II, we calculate the marginal probability of \(F\) by summing all the possible combinations from the probability distribution table where \(F\) is true (i.e. the probability of manipulating a file is true). Using the chain rule of probability, these joint probabilities are then expanded into conditional probabilities to smooth the computation process. The procedure for calculating the marginal probability also known as normalized probability or data evidence is shown in Table III.

The above calculations for evaluating a normalized probability for \(F\) indicate that the chance of a file being manipulated is 0.31 in the
absence of any other evidence. Now, if we know that a file $F$ has been manipulated (i.e. $F$ is true), then we can calculate the posterior probabilities of the application programs $A_1$, $A_2$, and $A_3$ by using the Bayes theorem from the initial probabilities calculated previously. This procedure, described in Table IV, indicates which of the application programs $A_1$, $A_2$, or $A_3$ manipulated the file $F$.

Since the posterior probability of application program $A_3$ is greater than those of $A_1$ and $A_2$, we can conclude that the file $F$ is more likely to be manipulated by the application program $A_3$. It is worth mentioning that since the difference among the probabilities is distinct, we single out the application program $A_3$. In the case of two or more probabilities being close in value, we can conclude that two or more application programs caused the file manipulations, one after another.

V. Results and Discussion

The matter of obtaining the application program's accurate footprint is complicated by the fact that much of the application footprint is modified each time the application runs. Although it is easy to find evidence of the last time that an application ran (e.g. the last access time on the application binary), it is very strenuous and laborious to find shadowed application runs. Evidence of these runs must be derived from events within the audit log files, history files and temporary files found either within the filesystem entries or by searching for their traces in the free blocks of the storage media. As one retreats in time, the evidence for these application runs becomes less reliable, as memory blocks within the free list are reallocated to other files and are overwritten according to the operating system's disk allocation policy. Eventually, in such scenarios of uncertainty, we propose to attach a likelihood measure to the assertion that an application was running at that particular time. Bayesian networks provide a solid and efficient way to represent joint distributions for a large number of random variables, and allow effective inference from the observations. Therefore, they can be used to examine and learn probabilistic and causal relationships by updating beliefs based on the evidence provided. The issue of handling the uncertainties inherent in filesystem activities of the application runs prompted us to use Bayesian networks in our project.

Our methodology consists of the following five steps:

- data collection
- data preprocessing
- hypothesis generation
- experimentation
- model evaluation

We conducted experiments on filesystem datasets obtained by our indigenously developed program in C#. It is very important to select the right data structure when designing a network classifier, as adequately preprocessed data structure leads to improved performance for Bayesian classifier learning. The basis of our hypothesis generation was that the filesystem manipulation dataset was drawn from a random sample space. For this purpose, we decided to use the Bayesian approach to classify the dataset based on our background knowledge about the filesystem access patterns of different applications. Bayesian learning algorithms, such as the naive Bayes' classifier, provide a practical approach to learning problems, and produce promising results compared to decision tree and neural network algorithms. In particular, Bayesian learning algorithms are effective especially for classifying data that involves text attributes [5, 20].

Our model demonstrates how Bayesian networks can be used effectively as a predictive tool for assessing the verification application program executions from the filesystem metadata information. Our Bayesian network model for timeline reconstruction of the filesystem activity can cope with the large complexity of datasets in a reasonable timeframe, and is helpful in enhancing the digital forensic analysis procedure with improved performance. The network struc-
ture was specified manually using knowledge about how the application programs affect the filesystem attributes. We used a Bayesian scoring method to assign a probability score for a network model. However, it is quite likely that the manual design may introduce bias and would not be applicable for more complicated data structures with a large number of added features.

An important feature of a forensic analysis software system is its ability to check the false alarm rate. To ensure accuracy of the results, the network should correctly identify normal and abnormal behaviours. In the forensic literature, the majority of researchers emphasize the importance of addressing the false positive rate. However, we believe that handling the false negatives is equally important in digital forensic analysis because the production of false negative results in overlooking some valuable filesystem state information that may have a greater impact on obtaining digital evidence.

For this purpose, we conducted two types of evaluations to detect false positives and negatives. False positive (type-I error) tests were conducted by feeding perfect instances of directories and files that were known not to have been manipulated by an application program. The number of cases wrongly classified by the network as known cases were treated as false positives. False negative (type-II errors) tests were conducted by supplying the network with the known perfect instances of directories and files identified to be accessed by an application program. The number of cases wrongly classified by the network as unknown cases were treated as false negatives. We calculated the ratios of false positives to false negatives, with respect to the total number of cases. False positives and negatives were calculated to be 6.25% and 4.85%, respectively, for the final experimental dataset. A graph showing the percentage of false positives and false negatives observed for ten different Bayesian networks trained over increasing sizes of datasets is provided in Figure 4.

![False Positives and False Negatives Statistics](image)

Figure 4: Graph showing the percentage of false positives (FP) and false negatives (FN) observed for ten different networks trained over increasing sizes of datasets.

The graph shows that sufficiently larger training datasets resulted in reduced false findings (positives and negatives). This observation is due to the fact that the larger number of observations pertaining to the filesystem activity patterns of the application programs help to calculate more accurate marginal and joint probability distribution tables, which are subsequently helpful in calculating more precise posterior probabilities.
It is important to mention that in data classification problems, the availability of large numbers of datasets improve one’s confidence in the results. Although we have collected data for a variety of different application program types, we still have a small number of datasets. Each individual dataset carries too much weight in the final findings, and adding results for more datasets would improve the overall network’s results.

The results that we have obtained are promising, as the total percentage of both the false positives and negatives was reduced to 11.10% when tested on the sensible size of the trained network. The added advantage of the Bayesian network, as compared to the ordinary neural network, is that it avoids over-fitting and is thus more flexible. Finally, we executed SQL queries on the filesystem activity dataset for cross verification of the accuracy of the results.

To the best of our knowledge, the approach that we have proposed herein to learn the application programs’ execution pattern using Bayesian belief networks has not been attempted by any other researcher, and has not appeared in any seminal paper.

VI. Conclusions

In this paper, we presented an efficient and adaptable Bayesian network–based probabilistic framework with which to identify files that have been manipulated by different applications. Our method could be useful for preparing a concise timeline of various activities occurring on a filesystem. The focus of this paper is the learning of a model from sample data that can be used subsequently for analyzing a filesystem state to determine the previously executed application programs. We used the layers of the abstraction approach to obtain, correlate, preprocess, train and test the filesystem datasets. The efficacy of this research is to determine the model parameters that maximize the supervised conditional likelihood, rather than the commonly used unsupervised likelihood methods. A probabilistic framework for this project was developed, and experiments were conducted using different applications to extract prior knowledge about the pattern of filesystem manipulations. This prior knowledge was used to estimate the posterior information from the test dataset. However, a challenge associated with our technique is the difficulty of obtaining a perfect footprint of the execution pattern of various application programs.

VII. Future Directions

The practical difficulty in applying Bayesian methods is that they typically require initial knowledge of marginal and joint distributions of the probabilities about different attributes of the dataset. When these probabilities are not known in advance, they are simply estimated based on background knowledge of the dataset, or previously available data and assumptions about the form of the underlying probability distributions. Specifically, this task becomes more challenging when the forensic investigation requires gathering evidence from disparate sources. Another practical difficulty is the significant computational investment required to choose the optimal Bayes’ hypothesis and select a suitable prior distribution to maximize the posterior probability. A constraint to our approach is that the overhead associated with the calculation of prior probability for a larger number of applications is an important consideration. However, we believe that using more advanced statistical analysis techniques that support the training data in an unsupervised mode could enhance network performance. The key issue in our research is the development of a mechanism that supports automatic input parameter learning from a large dataset. For automatic input parameter learning, the expectation maximization algorithms, such as maximum likelihood and maximum a posteriori could be attempted as a future dimension to
this research to increase the robustness of the analysis procedure.

References


TABLE I
PRIOR PROBABILITIES OF MANIPULATING A FILE BY THREE APPLICATIONS A₁, A₂ AND A₃

<table>
<thead>
<tr>
<th></th>
<th>Probability of manipulation by Application 1 (A₁)</th>
<th>Probability of manipulation by Application 2 (A₂)</th>
<th>Probability of manipulation by Application 3 (A₃)</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>0.25</td>
<td>0.34</td>
<td>0.40</td>
</tr>
<tr>
<td>False</td>
<td>0.75</td>
<td>0.66</td>
<td>0.60</td>
</tr>
</tbody>
</table>

TABLE II
JOINT PROBABILITY DISTRIBUTION TABLE FOR MANIPULATION OF A FILE BY THREE APPLICATION PROGRAMS, A₁, A₂ AND A₃

<table>
<thead>
<tr>
<th>A₁</th>
<th>True</th>
<th>False</th>
<th>True</th>
<th>False</th>
<th>True</th>
<th>False</th>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>True (file)</td>
<td>p(F</td>
<td>A₁</td>
<td>A₂ A₃)</td>
<td>p(F</td>
<td>A₁</td>
<td>A₂' A₃)</td>
<td>p(F</td>
<td>A₁</td>
</tr>
<tr>
<td></td>
<td>0.88</td>
<td>0.46</td>
<td>0.51</td>
<td>0.25</td>
<td>0.62</td>
<td>0.34</td>
<td>0.40</td>
<td>0.00</td>
</tr>
<tr>
<td>False (file)</td>
<td>p(F'</td>
<td>A₁</td>
<td>A₂ A₃)</td>
<td>p(F'</td>
<td>A₁</td>
<td>A₂' A₃)</td>
<td>p(F'</td>
<td>A₁</td>
</tr>
<tr>
<td></td>
<td>0.12</td>
<td>0.54</td>
<td>0.49</td>
<td>0.75</td>
<td>0.38</td>
<td>0.66</td>
<td>0.60</td>
<td>1.00</td>
</tr>
</tbody>
</table>

TABLE III
OUTLINE OF THE PROCEDURE FOR CALCULATING THE MARGINAL PROBABILITY OF THE FILE (F)

\[
p(F) = p(F | A₁ A₂ A₃) + p(F | A₁ A₂' A₃) + p(F | A₁ A'₂ A₃) + p(F | A₁ A'₁ A₂ A₃) + p(F | A₁ A'₁ A₂' A₃) + p(F | A₁ A'₁ A'₂ A₃) + p(F | A₂ A₁ A₃) + p(F | A₂ A₁' A₃) + p(F | A₂ A'₁ A₃) + p(F | A₂ A'₁ A'₃) + p(F | A₂' A₁ A₃) + p(F | A₂' A₁ A'₃) + p(F | A₂' A₁' A₃) + p(F | A₂' A₁' A'₃) + p(F | A₃ A₁ A₂) + p(F | A₃ A₁' A₂) + p(F | A₃ A'₁ A₂) + p(F | A₃ A'₁ A'₂) + p(F | A₃ A'₁' A₂) + p(F | A₃ A'₁' A'₂) + p(F | A₃ A'₁' A'₃) + p(F | A₃ A'₁' A'₃)
\]

\[
= p(F | A₁ A₂ A₃) \times p(A₁) \times p(A₂) \times p(A₃) + p(F | A₁ A₂' A₃) \times p(A₁) \times p(A₂') \times p(A₃) + p(F | A₁ A'₂ A₃) \times p(A₁) \times p(A'₂) \times p(A₃) + p(F | A₁ A'₁ A₂ A₃) \times p(A₁) \times p(A₂) \times p(A'₃) + p(F | A₁ A'₁ A₂' A₃) \times p(A₁) \times p(A₂') \times p(A'₃) + p(F | A₁ A'₁ A'₂ A₃) \times p(A₁) \times p(A'₂) \times p(A'₃) + p(F | A₂ A₁ A₃) \times p(A₂) \times p(A₁) \times p(A₃) + p(F | A₂ A₁' A₃) \times p(A₂) \times p(A₁') \times p(A₃) + p(F | A₂ A'₁ A₃) \times p(A₂) \times p(A'₁) \times p(A₃) + p(F | A₂ A'₁ A'₃) \times p(A₂) \times p(A'₁) \times p(A'₃) + p(F | A₂' A₁ A₃) \times p(A₂') \times p(A₁) \times p(A₃) + p(F | A₂' A₁ A'₃) \times p(A₂') \times p(A₁') \times p(A₃) + p(F | A₂' A₁' A₃) \times p(A₂') \times p(A₁') \times p(A₃) + p(F | A₂' A₁' A'₃) \times p(A₂') \times p(A₁') \times p(A'₃)
\]

\[
= 0.88 \times 0.25 \times 0.34 \times 0.40 + 0.62 \times 0.25 \times 0.34 \times 0.40 + 0.51 \times 0.25 \times 0.34 \times 0.40 + 0.46 \times 0.25 \times 0.34 \times 0.60 + 0.40 \times 0.75 \times 0.34 \times 0.40 + 0.34 \times 0.75 \times 0.34 \times 0.60 + 0.25 \times 0.25 \times 0.66 \times 0.40 + 0.00 \times 0.25 \times 0.66 \times 0.60
\]

\[
= 0.30625 = 0.31
\]
### TABLE III

**ILLUSTRATION OF THE PROCEDURE FOR CALCULATING THE POSTERIOR PROBABILITIES OF APPLICATION PROGRAMS A₁, A₂ AND A₃ BY USING THE BAYES THEOREM**

<table>
<thead>
<tr>
<th>Probability Calculation</th>
<th>Equation</th>
<th>Result</th>
</tr>
</thead>
</table>
| P (A₁ | F= true)              | \[
\begin{align*}
p(A₁ | F= \text{true}) &= \frac{p(F | A₁) x p(A₁)}{p(F)} \\
&= \left[ \frac{p(F | A₁) x p(A₁) + p(F | A₁ A₂) x p(A₂) + p(F | A₁ A₂ A₃) x p(A₃)}{p(F)} \right] / p(F) \\
&= \left[ \frac{10.88 x 0.34 x 0.40 + 0.46 x 0.34 x 0.60 + 0.51 x 0.66 x 0.40 + 0.25 x 0.66 x 0.60}{0.30625} \right] / 0.30625 \\
&= \frac{0.36503}{0.30625} \\
&= 0.37
\end{align*}
\] | 0.37 |
| P (A₂ | F= true)              | \[
\begin{align*}
p(A₂ | F= \text{true}) &= \frac{p(F | A₂) x p(A₂)}{p(F)} \\
&= \left[ \frac{p(F | A₂) x p(A₂) + p(F | A₂ A₃) x p(A₃) + p(F | A₂ A₃ A₄) x p(A₄)}{p(F)} \right] / p(F) \\
&= \left[ \frac{10.88 x 0.25 x 0.40 + 0.46 x 0.25 x 0.60 + 0.62 x 0.75 x 0.40 + 0.34 x 0.75 x 0.60}{0.30625} \right] / 0.30625 \\
&= \frac{0.55066}{0.30625} \\
&= 0.55
\end{align*}
\] | 0.55 |
| P (A₃ | F= true)              | \[
\begin{align*}
p(A₃ | F= \text{true}) &= \frac{p(F | A₃) x p(A₃)}{p(F)} \\
&= \left[ \frac{p(F | A₃) x p(A₃) + p(F | A₃ A₄) x p(A₄) + p(F | A₃ A₄ A₅) x p(A₅)}{p(F)} \right] / p(F) \\
&= \left[ \frac{0.88 x 0.25 x 0.34 + 0.51 x 0.25 x 0.66 + 0.82 x 0.75 x 0.34 + 0.41 x 0.75 x 0.66}{0.30625} \right] / 0.30625 \\
&= \frac{0.672 72}{0.30625} \\
&= 0.67
\end{align*}
\] | 0.67 |
Muhammad Naeem Khan graduated from Quaid-i-Azam University, Islamabad, Pakistan with a M.Sc. in Computer Science. He is currently pursuing his D.Phil. in Engineering at the University of Sussex, Brighton, UK. His research interest is on the application of machine learning techniques for digital forensic analysis.

Professor Chris Chatwin holds the Chair of Industrial Informatics and Manufacturing Systems (IIMS) at the University of Sussex, UK. He has published two research level books: one on numerical methods, the other on hybrid optical/digital computing - and more than two hundred international papers which focus on: optics, optical computing, signal processing, optical filtering, laser physics, beam/target interactions, knowledge based control systems, expert systems, computer integrated manufacture, manufacturing communication systems, rapid prototyping, micro-engineering, computational numerical methods, instrumentation, digital image processing, intelligent digital control systems, digital electronics, wireless communication systems, wireless networks, e-commerce, secure communications and computer forensics. Professor Chatwin is on the editorial board of the International Journal "Lasers in Engineering". He is also a member of: the Institution of Electrical and Electronic Engineers; the British Computer Society; the Association of Industrial Laser Users. He is a Chartered Engineer, Euro-Engineer, International Professional Engineer, Chartered Physicist, Chartered Scientist and a Fellow of: The Institution of Electrical Engineers, The Institution of Mechanical Engineers, The Institute of Physics and The Royal Society for Arts, Manufacture and Commerce.

Dr Rupert Young is a Reader in the Department of Engineering at University of Sussex. His research interests include pattern recognition, image processing, machine vision and digital signal analysis techniques. He is author of over 80 refereed journal papers. He is a regular reviewer for Optics Letters, Applied Optics, Electronics Letters and Optical Engineering journals. He is also a member of the Institute of Electronic and Electrical Engineers, the Society of Photo-Optical & Instrumentation Engineers and the Optical Society of America. He chairs sessions of the Optical Pattern Recognition Conference at the annual SPIE AeroSense Conference in Orlando, Florida.