Identity Application Fraud Detection using Web Mining and Rule-based Decision Tree

Amany Abdelhalim[1] and Issa Traore[2]

[1]Department of Electrical and Computer Engineering
University of Victoria, P.O. Box 3055 STN CSC,
Victoria, B.C., V8W 3P6, Canada
Email: amany@ece.uvic.ca

[2]Department of Electrical and Computer Engineering
University of Victoria, P.O. Box 3055 STN CSC,
Victoria, B.C., V8W 3P6, Canada
Email: itraore@ece.uvic.ca

Abstract: ***Identity fraud is becoming a growing concern for most government and private institutions. In the literature, identity frauds are categorized into two classes, namely application fraud and behavioural (or transactional) fraud. Most of the previous works in the area of identity fraud prevention and detection have focused primarily on credit transactional frauds. The work described in this paper is one of the very few works that focus on application fraud detection. We present an unsupervised framework to detect fraudulent applications for identity certificates by extracting identity patterns from the web, and crossing these patterns with information contained in the application forms in order to detect inconsistencies or anomalies. The outcome of this process is submitted to a decision tree classifier generated on the fly from a rule base which is derived from heuristics and expert knowledge, and updated as more information is obtained on fraudulent behaviour. We evaluate the proposed framework by collecting real identity information online and generating synthetic fraud cases.***

Keywords: Application fraud, Fraud detection, Anomaly detection, Web mining, Rule-based Decision Tree.

1. INTRODUCTION

Identity fraud is spreading fast and causing more and more damages both financially and sociologically. Identity fraud occurs when a criminal impersonates another individual by taking on that person's identity or by creating a fake identity for whatever reason [1] and [2].

Identity frauds can be categorized into two different types: transaction frauds and application frauds. Application fraud occurs when an individual or an organization applies for an identity certificate (e.g., passport, credit card etc.) using someone else's identity. Transaction fraud, also known as behavioral fraud, occurs when an identity thief performs some operations or transactions using fake or stolen identity. Most of the research in identity fraud detection has focused so far on credit transactional fraud detection. Limited attention has been paid to application fraud detection, where only few papers have been published so far. Application fraud detection, however, is an important aspect of any sound and global strategy to combat identity fraud. Application fraud detection is a proactive measure that allows early screening of fraudsters, contributing as a result to cutting down significantly the effort and resources required to detect fraudulent transactions [3].

Application frauds share many of the characteristics of transaction frauds, including the large amount of data processed, the need to make an acceptance or rejection decision in a short time span, and the significant imbalance between normal and fraudulent data (fraud cases represent only a very small fraction of performed operations). These issues have widely been studied in the transaction fraud literature. The open issues concern the differences between application and transaction frauds.

Transactional operations benefit typically from significant historical records collected over time for a single person. Many Artificial Intelligence (AI) techniques can be used to process past usage history and provide fairly accurate fraud detection results. In contrast historical information related to application operations for a specific person. is in general very limited if non existent. For instance, there is a significant delay between consecutive applications for passport renewal in most countries; in some countries it takes between 5 to 10 years before having to renew a passport. As a result the background historical data per individual is very limited. Furthermore the application file contains only sparse identity attributes, which by themselves might not be enough to make a proper decision. So application fraud detection requires specific analysis techniques which can compensate for the weak characteristics of the available data.

Another key difference between transaction and application fraud data is the breadth of the identity information covered. While as discussed above, transaction fraud data involve significant depth (at least from historical standpoint), application fraud data have wider scope. Application forms may potentially be linked to a wider variety of data sources than what typical transactional operations could enjoy. For instance, information contained in a passport application may be crossed with collateral information in many other databases like vital statistics, death records, social security database, birth records etc. In contrast transactional operations typically cover only a limited number of different

data sources maintained internally by the issuer or available externally. For instance, for legal and competitive reasons, credit card transactions can be cross-checked against only few external databases such as credit bureaus.

An alternative identity information source that transcends the above restriction is the web. The web is actually a federation of many different identity information data sources. Using the web, it may be possible to cross-check application forms with data from various collateral identity information sources; even though such information might be sparse or useful information could be missing.

The web is one of the richest and diverse sources of identity information. But at the same time it is one of the most challenging to deal with because of the unstructured nature of the data involved. To our knowledge none of the application fraud detection frameworks proposed so far has explored the strength of such data source. All of them use traditional offline or private data sources. In many ways, the Internet serves as a key vehicle for identity fraud. The Internet represents an appealing place for fraudsters to collect a host of personal and financial data related to many innocent users. Using the collected data they can impersonate the users and commit fraudulent activities using stolen or fake identities. Mining Internet data for fraudulent purposes using a search engine is commonly referred to as *black hat Google hacking.*

In this work, we propose an application fraud detection framework that consists of two main components: an online identity mining module and a fraud detector. Our proposed online identity mining scheme is based on *white hat Google hacking*, in which identity information is collected through online search, targeting a specific individual (i.e. the applicant). The fraud detector is an intelligent unsupervised decision model that analyzes extracted online identity information related to the applicant and crosses such information with information contained in the application form, in order to detect and report possible inconsistencies or anomalies. We designed our fraud detector using a rule-based decision tree technique fed with a set of simple heuristics that define general and common understanding of the notion of fraudulent and normal behaviours.

Although various machine learning techniques (supervised or unsupervised) may be used in designing such kind of detector, the lack of genuine fraud data tends to hinder such process. A common challenge of data-mining based fraud detection research is the lack of publicly available real data for model building and evaluation. For privacy and competitive reasons, organizations are reluctant to release data related to fraudulent activities.

In order to compensate for the fact that labels may not be readily available, ideally such techniques should be unsupervised. Although many unsupervised detection frameworks have so far been proposed for transaction fraud detection, none of them apply specifically to application fraud detection. To our knowledge, the application fraud detection techniques proposed so far in the research literature are either supervised or semi-supervised.

The solution commonly adopted in the industry and in the literature to address this issue is to encode expert knowledge and past knowledge of fraudulent behavior into rule bases. Likewise, the application fraud detection techniques used in commercial applications and in the research literature include rule-based matching of credit application and credit history, fraud matching using black lists (based on previous fraudulent applications), and supervised model based on labeled data. However, due to the fast pace at which new fraud methods are created and used by fraudsters, the rule bases are submitted to constant changes and usually tend to grow at an accelerated rhythm, quickly reaching unmanageable size. This might not be conducive to timely decision, which is required in many business environments. In this case, a decision tree represents an effective alternative to rule-base reasoning for a quicker decision. This is because in order to be able to make a decision for some situation we need to decide the shortest and most efficient order in which tests should be evaluated. In that case a decision structure (e.g. decision tree) is much quicker than a rule engine to reach a decision. In the decision tree the order of checking conditions and executing actions is immediately noticeable. Second, conditions and actions of decision trees are found on some branches but not on others. Those conditions and actions that are critical are connected directly to other conditions and actions, whereas those conditions that do not matter are absent. In other words it does not have to be symmetrical. Third, decision trees are more readily understood by others in the organization than a set of rules. Consequently, they are more appropriate as a communication tool.

Decision trees can be an effective tool for guiding a decision process as long as no changes occur in the conditions or dataset used to create the decision tree. When the conditions change, as this happens constantly with fraud cases, restructuring the decision tree becomes a desirable task. It is difficult, however, to manipulate or restructure decision trees. This is because a decision tree is a procedural knowledge representation, which imposes an evaluation order on the attributes. In contrast, a declarative representation, such as a set of decision rules is much easier to modify and adapt to different situations than a procedural one. This easiness is due to the absence of constraints on the order of evaluating the rules [4].

Rule-based decision tree techniques bridge the divide between rule-base systems and decision trees, by allowing the on-demand creation of a short and accurate decision tree from a stable or dynamically changing set of rules. On one hand decision tree paradigm easily allows changes to the conditions (when needed) by modifying the rules rather than the decision tree itself. On the other hand the paradigm takes advantage of the structure of the decision tree to organize the rules in a concise and efficient way required to take the best decision. So knowledge can be stored in a declarative rule form and then be transformed (on the fly) into a decision tree only when needed for a decision making situation [4] using a rule-based decision tree technique. We use *RBDT-1*, a rule-based decision tree technique recently proposed in [5] to design our fraud detector. *RBDT-1* has been shown to be more effective in terms of tree complexity than other existing similar techniques [6].

We evaluate the proposed framework using a database of real online identity information collected through white hat Google hacking combined with a synthetic sample of fraud data that we created.

The rest of the paper is structured as follows. In Section 2, we summarize and discuss related work. In Section 3, we present our approach and give an overview of the general architecture of the system. In Section 4, we present our online identity information retrieval scheme. In Section 5, we present our identity fraud detection scheme. In Section 6, we conduct the evaluation of the proposed framework. Finally in Section 7, we make some concluding remarks.

2. RELATED WORK

Although there is a large amount of published works on identity fraud detection, only a few of these works focus specifically on application fraud detection. We review in this section representative samples of these works.

Wang *et al.* developed a general framework to analyze identity theft in the context of integrated multiparty perspectives [2]. Although their work is very important, it is mostly conceptual and focuses on identifying the stakeholders involved in identity fraud detection and protection, and on characterizing the interactions between them. According to their framework, there are five main stakeholders: identity owner, identity protector, identity issuer, identity checker, and identity thief. Although we use some of the concepts defined in their framework, our work is different from their work in the sense that we focus primarily on studying, implementing and evaluating a concrete and practical identity checker.

Burns and Stanley in [7] discuss the techniques used by credit card issuers to screen credit applications. Card issuers use multiple data sources (e.g., credit bureaus) to confirm the information listed in an application form. They monitor relevant databases for any changes to the consumer's credit records, personal address or phone number, which often give the earliest indication of identity theft. They also calculate spatial information such as the distance between the phone number and address presented in the application to determine if they originate from the same area code or not. Applications which are submitted from areas where a lot of fraud cases appeared before are also reviewed thoroughly.

Cross-referencing new applications with similar information from other databases is a common characteristic of most of the application fraud detection approaches proposed so far in the literature, including [8], [9] and [10].

Wheeler and Aitken in [8] applied case-based reasoning for credit card application fraud detection. The proposed system was used as reinforcement for an existing rule-based (RB) fraud detector. The input data to the system consists of pairs of database records, consisting in one hand of the application and on the other hand of fraud evidence produced by the RB system. The goal is to reduce the fraud investigations by combining the diagnosis of multiple algorithms in producing the final decision. The proposed case based reasoning framework consists of two decision-making modules, in charge of case retrieval and diagnosis, respectively. The retrieval component utilizes a weighting matrix and nearest neighbor matching to identify and extract appropriate cases to be used in the final diagnosis for fraud. The decision component utilizes a set of algorithms (i.e. probabilistic curve selection, best match algorithm, negative selection algorithm, density selection algorithm, and default goal) to analyze the retrieved cases and attempt to reach a final diagnosis. The results of the system showed that the performance of each algorithm employed in the fraud diagnosis process differs depending on the nature of the fraud case presented. In our work we also use application cross-referencing to capture similarities. However, our fraud detection algorithm does not need labels to make fraud or non-fraud decisions. While in the work of Wheeler and Aitken, the evidence has to be produced and tagged as fraud or non-fraud by a separate system, our proposed system uses unlabelled data in its decision-making process.

Phua *et al.* in [9] proposed a technique for detecting application fraud based on implicit links between new and previous applications. Using a communal suspicion scoring scheme, they classify a new application as belonging to one of three lists, a black list, a white list or an anomalous list. They performed the classification by finding a match between information contained in a new application and information from an existing application in one of the lists. Both the information in the new application and the corresponding linked application were represented by an attribute vector. The suspicious score is the summation of each pair-wise attributes which is expressed by either exact matching or fuzzy matching. They also assigned weights to the attributes depending on their nature. The proposed technique is similar to the weight matrix proposed in [8], except that in [9], in addition to basing their calculation of suspicion scores on matching attributes, they take into consideration temporal and spatial differences between the matching applications. A key difference between this approach and ours is that it requires labeling the data. Despite this important difference, we evaluate our work by mainly comparing it to the approach proposed by Phua *et al.*, as explained later.

In [10], a system that detects subscription fraud in fixed telecommunications was proposed. The system consisted of two main modules, a classification module and a prediction module. The purpose of the prediction module was to detect fraudulent customers when they attempt to subscribe to a fixed telecommunication service. To investigate the application for signs of fraud, the prediction module crosses the information available in the new application with information available in the account database. This allows linking the new application with existing fraudulent accounts. Another source of information that was used was a public database that lists situations of insolvency mostly related to banks and department stores. The prediction module consisted of a multi-layer feed-forward neural network. The output units indicated one of two decisions for an application; either fraudulent or legitimate. A dataset of subscription examples was labeled and split into the following three sets: a training set, a validation set, and a testing set. From their experiments they outlined that the prediction module was able to identify only 56.2% of the true fraud cases while screening 3.5% of all the subscribers in the testing set. Like in [8] and [9], the proposed framework requires using labeled data and some form of supervision. Many criticisms can be made about using labeled data for fraud detection such as the limited efficiency of processing such data in event-driven environment, the cost, difficulty, and length of time needed to obtain such data, and the fact that the class labels can be inaccurate. In

our work, we do not use or assume knowledge of existing fraudulent applications in the screening process. Instead, our proposed fraud detector processes unlabelled data from the identity information source.

In the Identity Angel approach proposed by Sweeney, identity information is extracted from the Web [11] and [12]. The goal of the Identity Angel project is to locate online resumes that contain sufficient information for a criminal to acquire a new credit card in the name of the owner of the resume, and then to notify the corresponding subject by sending him an email, encouraging him to remove such sensitive information.

Identity Angel is implemented using a Java program that uses filtered search and the Google API to identify resumes, and uses entity detectors for SSNs, dates of birth, and email addresses to extract information from online resumes. While the identity information retrieval technique used by Sweeney is closely related to ours, an important difference is that the search space used in the identity angel project is limited to a specific kind of online documents, which are online resumes. In our case, our search space is the entire web and targets any kind of documents, which gives a wider scope for our knowledge source. Furthermore, the identity angel tool focuses only on information retrieval and does not implement any formal fraud detection process or algorithm.

3. CONCEPTUAL FRAMEWORK AND ARCHITECTURE

In this section, we describe basic identity concepts and give an outline of our identity fraud detection framework design.

3.1 Identity Concepts

Identity can be defined "as one or more pieces of information that cause others to believe they know who someone is" [13]. We divide such pieces of information into two categories: basic identity information and specific identity information. Basic identity information simply refers to the first name and last name of an individual. Specific identity information refers to additional identifying information other than the basic identity information, such as birth date, social security number, address, credit card number, etc. A targeted search strategy typically consists of using basic identity information to obtain specific identity information for a particular individual.

In [2], the notion of identity certificate is introduced. An identity owner applies for an identity certificate for various purposes in his life, either administrative or business related. Examples of identity certificates include passport, social security card, driver's license, health card, credit card, digital (security) certificate, etc. Identity issuers, represented by trusted government or private institutions, deliver identity certificates.

An identity certificate usually includes at least one or several of the following pieces of identity information: certificate number, owner's information, the purpose of the certificate (social security or credit card), issuer's information, validity time period, issuer's certification, and owner security information such as his signature or the security number used for credit card. An identity owner is usually characterized by providing at least one or several of the following pieces of identity information: the first and last names, the parents' names with a particular emphasis on mother's maiden name, the address, and the date and place of birth.

Although identity issuers have various and more or less sophisticated ways to check the validity of identity certificates, identity certificates still represent the main vehicle used to conduct identity fraud. The issuing and use of identity certificates rely on a chain of trust. For instance, the issuing of a credit card relies on the social security card, which in its turn relies on the passport, which again relies on the birth certificate. This chain of trust can be broken, when a fraudster is able to steal one's identity or to fake a new one.

To obtain an identity certificate an individual submits an *application* to a certificate issuer providing required identification information. In doing so, she makes an *identity claim.* So, an *identity claim* occurs when an individual declares a specific identity, for instance, on an official document like a passport application, or a business document like a credit card application.

3.2 General Architecture

Our application fraud detection framework consists of three main modules as illustrated by Figure 1: the identity information source, the identity information retrieval engine, and a fraud detector. The identity information source can be a single or a combination of several identity data sources such as credit bureaus databases, previous applications databases, vital statistics, or the web.

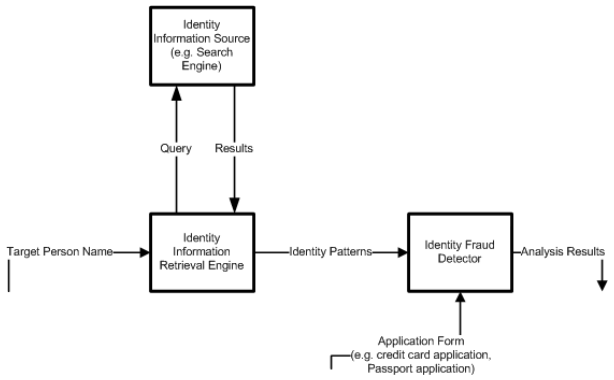
The identity information retrieval engine queries the identity information source by targeting the identity applicant, and feeds the targeted search results into the identity fraud detector. The identity fraud detector crosses and analyzes, using heuristics, the search results with background identity information fetched from the application file (e.g., credit card application) being checked.

The identity information retrieved for a particular individual consists of a collection of identity information patterns. Each pattern may consist of a sequence of identity information pieces such as address, credit card number, mother maiden name, social security number etc. Resulting from a targeted search process, the patterns may belong to different individuals that simply happen to share the same name. So it is necessary to develop a strategy to sort and narrow down the returned patterns for the target individual. The sorted information can then be analyzed to detect possible identity fraud.

Identity fraud detection will consist of screening a particular identity claim for possible inconsistencies with the information in the search results. The system will extract the identity information for the individual found in the search results and check the results against the identity claim, reporting any discrepancy as an anomaly. This may either lead to a rejection of the application or to further investigations with some follow-up questions.

We will describe in subsequent sections the identity information retrieval engine and the identity fraud detector.

Figure 1. Fraud Detection Framework



4. WEB-BASED IDENTITY INFORMATION RETRIEVAL ENGINE

The design of the identity information retrieval engine depends on the identity information source. Unlike with other identity information sources, identity information retrieval on the web involves a lot of challenges because of the scale, complexity, and diversity of the information provided. On the other hand, the web represents a powerful source of identity information that can be used effectively in combating identity fraud. We describe in this section the challenges and algorithms involved in the design of our web-based identity information retrieval engine.

4.1 Design challenges and strategies

Although many tools are readily available for information retrieval, identity information retrieval based on targeted search is a challenging task. Because of the huge amount of information available online, locating and sorting identity information related to a specific individual is a daunting task. We discuss in the following paragraphs some of the challenges involved in this task.

A key challenge is that a wide variety of document formats are used on the Internet (e.g., PDF, Doc, Excel, or Html). Some of these formats are not directly searchable, and require some form of pre-processing. Also our search space is not limited to a specific type of documents. As a matter of fact searching and extracting identity information from a document, which has only identity values for the target name such as the curriculum vitae is easier than doing the same with a document that could have more than one person's identity information such as a company employee list. For the latter, the challenge is to avoid extracting identity information for someone different from our target.

Furthermore it is difficult to establish the existence of values corresponding to the identity keywords located in the documents. The reason for such difficulty is that although some of the identity keywords targeted in our search have a fixed number of characters, such as social security number or telephone number, no standard format is used to express corresponding values in the documents on the Internet. So, it is necessary to design a general value filter that is capable of finding and extracting identity values based on the common ways that people usually tend to express them online. One of the proposed solutions is to limit the search area in the document starting from the target name and ending at the next existing name.

Another challenge is that some of our identity keywords have homonyms. In many cases, sorting information pertaining to different homonyms is challenging. In the literature, one of the main approaches used to handle homonyms consists of associating semantic information with the keyword. The semantic-based approach, however, does not make sense when the homonyms are proper nouns. Since we are dealing, in our work, with targeted search, homonyms based on proper nouns are very important. We handle this kind of homonyms using our identity profile derivation strategy, outlined in Section 5, which allows sorting and separating identity information belonging to different individuals who happen to be homonyms.

Besides that, each of the identity information pieces, searched for, has synonyms. Because our search space is the Internet, there is no standard word for expressing these keywords. So we have to come up with all the possible synonyms for each of the identity keywords and expand the search parameters in the identity keyword filter to include those synonyms.

Figure 2. Identity Information Detection (IID) Algorithm

Algorithm: *The Identity Information Detector IID*

Input: The target person's name *n*.

Output: A list of documents containing at least one value for any of the keywords detected, along with those keywords and their values.

Uses: Set of search keywords *Keys* and the web.

Steps:	
let D= ∅, a set that will contain all the documents returned from the search.	1
let D_t = ∅, a set that will contain all the documents returned from the search after converting them into text.	2
for each s ∈ Keys do:	3
D= SearchEngineApi (s+n)	4
endfor	5
for each d_i ∈ D do:	6
D_t= ConvertToText(d_i) ∪ D_t	7
endfor	8
for each d_i ∈ D_t do:	9
ArrayofKeys= KeywordFilter (d_i)	10
KeyValue = ValueFilter (d_t, ArrayofKeys)	11
if KeyValue ≠ ∅	12
result[i][1] = d_i	13
result [i][2] = KeyValue	14
endif	15
endfor	16
return result	17

4.2 Search algorithm

In order to illustrate our concepts, we limit (in our proof of concept) our search keywords to only credit card and social security information, which correspond to two of the most popular and sensitive identity certificates available online. We think that this does not affect in anyway the generality of our framework.

Our identity information retrieval scheme will use targeted search based on the following specific identity information:

- Social security number
- Date of birth
- Home address
- Home telephone number
- Mother maiden name
- Credit card number
- Credit card expiry date
- Credit card type
- Credit card security number

Table 1: List of Keywords for Automated Identity Information Retrieval

Identity Information	Search Keywords
Social security number	[ssn| social security number| social security no| social security #| ssn#| ssnum| ssno]
Date of birth	[date of birth| born| dob| birthplace and date| d.o.b| birthdate]
Home address	[address| add| home| home address]
Home phone number	[Phone | telephone| Phones| home |tel. | tel | ph]
Mother's maiden name	[mothers maiden name |mother's maiden name | mother maiden name | mmn]
Credit card number	[credit card number| creditno| creditnum| ccnum| ccno| cc#| card number| amex| master card]
Credit card expiry date	[credit expire date |expire date| e-date| expdate| expiration date]
Credit card type	[credit card type| card type| Ctype]
Credit card security number	[security number| verification number]

Through manual search using Google, different synonyms corresponding to the above identity information were learned and were used in implementing our identity information search module.

Table 1 depicts the selected keywords. All these keywords will be referred to, in the rest of the paper, as the identity search keywords.

Figure 2 depicts our identity information detection (IID) algorithm. The searching mechanism of the IID algorithm has three phases. The first phase is described in steps 1 through 8. During this phase, first of all the search engine's API is invoked to find web pages containing the name of the target user and at least one of the identity search keywords. In steps 6 and 8, the content of each document returned from the search is converted into text and saved in a text document. This is because the returned documents could be in different formats (e.g., PDF, DOC, XLS, HTML) and converting them into text enables us to easily parse them.

The second phase is described in steps 9 to 11. In this phase, a function named *KeywordFilter* described in Figure 3 is invoked in order to identify the exact list of identity search keywords appearing in each document. The keyword filter is designed while taking into consideration possible synonyms for each of the identity keywords described earlier.

The third phase covers steps 12 to 17. In this phase a (keyword) value filter function named *ValueFilter* described in Figure 4 is used to identify the exact list of values corresponding to the identity search keywords appearing in each document. The value filter is designed while taking into consideration all the different formats used to express the identity keyword values.

Finally, document names along with a list of identity keywords and values are displayed for only those documents that at least have one value corresponding to an identity keyword. The rest of the documents are discarded.

The prototype of our retrieval engine was implemented and tested using Google API. Figure 5 represents a screen shot of one of the files created by our tool as a result of searching for an individual named John Q. Jones. Each block of information corresponds to the identity information found in a single document for this particular individual. Each block of information corresponds to an identity information pattern, which as mentioned earlier will be processed (by the fraud detector) to detect possible inconsistencies with submitted applications by John Q. Jones.

Figure 3. Keyword filtering algorithm that uses a set of regular expressions to identify the identity keywords appearing in the document

Algorithm: *keywordFilter(d)*

Input: A document *d.*

Output: a list *Keys* of keywords found in the given document.

Uses: Set of regular expressions that returns a keyword if found or ∅ otherwise, and a list of keywords ArrayofK that we are going to search for in the document.

Steps:	
let ArrayofKeys = ∅, a placeholder for the list of keys found in the document	1
for each $k_i \in$ ArrayofK do:	2
KeyFound=KeyRegularExpression(d, k_i)	3
if KeyFound ≠ ∅	4
ArrayofKeys [i]=k_i	5
endif	6
endfor	7
return ArrayofKeys	8

5. IDENTITY FRAUD DETECTOR

We illustrate in this section our proposed identity fraud detection strategy and algorithms.

Our identity fraud detection approach consists of two major phases: the derivation of individual profiles and the analysis

of the derived profiles using rule-based decision tree. We illustrate each of the two phases in the following.

5.1 Shared identity information

Let P be the set of identity patterns returned by a targeted search. Each identity pattern $pi \in P$ is represented by a k-dimensional attribute vector $<p_{i1},..., p_{ik}>$. Possible examples of attributes include the following:

<Social security number, Date of birth, Address, Telephone number, Mother maiden name, Credit card number, Credit card expiry date, Credit card type, Credit card security number>

We exclude the name simply because at this stage we are dealing with identity information belonging to homonyms. A typical identity pattern returned from a targeted search might include only a subset of the k attributes set. The missing (or undefined) fields will simply be considered non-applicable (NA), and will not be used for the matching.

Figure 4. Value Filtering algorithm that uses a set of regular expressions to identify the values for the identity keywords appearing in the document

Algorithm: *ValueFilter(d, ArrayofKeys)* Input: A document *d*, along with the list of keywords ArrayofKeys found in that document. Output: An array of *m* rows and 2 columns, each row has one of the keywords found in the document along with its value, or an empty array otherwise if no keyword-values were found. Uses: Set of regular expressions that returns a value of a keyword if found or ∅ otherwise.	

Steps:
let ArrayofValues = ∅, a list that will contain the keywords found in the document along with their values.
 	
for each $k_i \in$ ArrayofKeys do:
ValueFound=ValueRegularExpression(d, k_i)
if ValueFound ≠ ∅
ArrayofValues [i][1]= k_i
ArrayofValues [i][2]= ValueFound
endif
endfor
return ArrayofValues
	1
	
	2
	3
	4
	5
	6
	7
	8
	9

Figure 6 illustrates sample identity patterns. In this example the following five identity attributes are considered: social security number (ssn), date of birth (dob), mother maiden name (mmn), address (addr) and phone number (tel).

The retrieval engine uses a name and a set of keywords to search the identity information source (e.g., the Web), and returns a collection of identity information related to the identity claim corresponding to the application being checked. As a result the returned identity information may correspond to several different individuals, all having the same name. For instance, we can have in the search result several social security numbers corresponding to the same name, which necessarily means that different individuals actually share the searched name. But this could also mean that the same person is impersonating all these different individuals by simply changing some of the identifying attributes. For the purpose of fraud detection, we need to identify and isolate the trail of identity patterns that relate to the individual submitting the application. We determine the patterns related to the applicant by determining direct and indirect connections between the application pattern and the patterns retrieved from the identity information source.

Figure 5. Sample search results returned by our tool for an individual named John Q. Jones. Each block of information represents an identity information pattern in our framework

```
JOHN Q. JONES

-----------------------------------------------
the keywords and their values found in the file named: html42.html
born
this could be a date of birth value 07 23 1832
1
-----------------------------------------------
the keywords and their values found in the file named: html43.html
born
this could be a date of birth value 03 23, 1944
1
-----------------------------------------------
the keywords and their values found in the file named: html44.html
0
-----------------------------------------------
the keywords and their values found in the file named: pdf25.txt
this could be a telephone number value (858) 534-3750
1
-----------------------------------------------
the keywords and their values found in the file named: pdf26.txt
0
-----------------------------------------------
the keywords and their values found in the file named: doc4.doc
social security number
this could be a social security number value 123-45-5678
date of birth
this could be a date of birth value 06/01/1993
home address
phone
this could be a telephone number value 410-272-1234
1
-----------------------------------------------
the keywords and their values found in the file named: html45.html
social security number
this could be a social security number value 123-45-6789
phone
this could be a telephone number value (410) 306-0229
1
-----------------------------------------------
the keywords and their values found in the file named: pdf27.txt
social security number
this could be a social security number value 123-45-6789
phone
this could be a telephone number value (410) 306-0229
1
-----------------------------------------------
the keywords and their values found in the file named: html46.html
ssn
this could be a social security number value 123456789
this could be a telephone number value 907-345-1234
1
```

Two patterns are directly connected if they share at least one attribute value. Two patterns p and q have an indirect connection, if there is a sequence of directly connected patterns linking them. In other words, there exists a sequence of patterns $p1 \ldots pn$, such that (p, p_1), (p_n,q), and (p_i, p_{i+1}), $\forall i \in \{1,\ldots,n-1\}$, are each a direct connection.

For the purpose of fraud detection, we trim the returned set P of identity patterns, and retain only the patterns having direct or indirect connections with the application pattern; let P^+ denote the subset of P containing all such patterns. Let p_0 denote the application pattern and let G denote a set of patterns such that $G = P^+ \cup \{p_0\}$.

As an example, let's consider the sample patterns depicted by Figure 6. Figure 7 depicts a tree structure exposing the direct and indirect connections between the sample patterns shown in Figure 7. We assume in this example that p_0 is the application pattern and the set of retrieved patterns is P, $P= \{p_1, p_2, p_3, p_4\}$. The set of connected patterns is G, $G= \{p_0, p_1, p_2, p_3\}$. For instance, patterns p_0 and p_2 share two

attributes, namely *ssn* and *mmn*. Pattern p_4 has no connection (direct or indirect) with p_0, so it is excluded from the fraud analysis. It is assumed in this case that p_4 does not belong to the applicant but belongs to another person that shares his name.

Figure 6. Examples of identity patterns based on 5 attributes

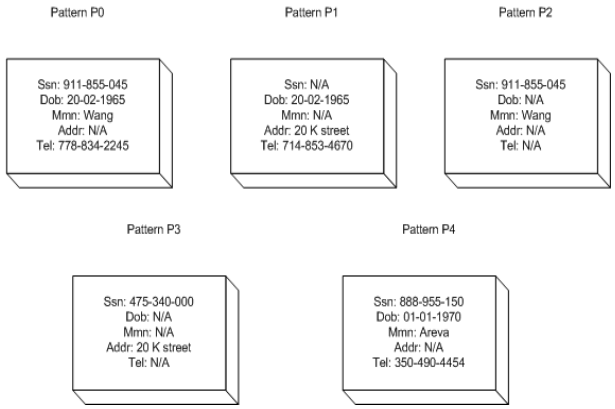
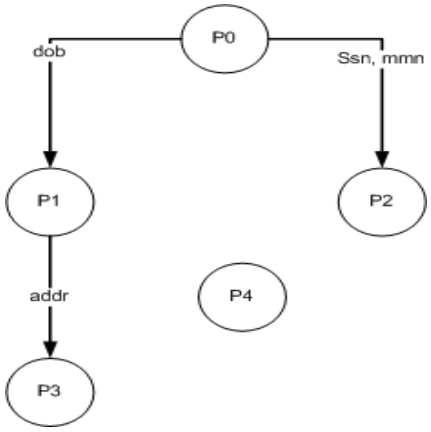


Figure 7. Direct and indirect connections between identity patterns



For the purpose of fraud detection, we analyze the link between every pattern-pair from *G* which are directly connected. Specifically we convert every pattern-pair that are directly connected into a single feature vector characterizing the underlying relationship. For each pattern-pair $< p_i, p_j > \in G \times G$, we derive a feature vector $v_{ij} = [\delta_{ijl}]_{1 \le l \le k}$, such that:

$$\delta_{ijl} = \begin{cases} ? \ if \ ((p_{il} = na) \ or \ (p_{il} = na)) \\ 1 \ if \ p_{il} = p_{jl} \\ 0 \ otherwise \end{cases} \quad (1)$$

Where "?" denote a missing value, which may be either 1 or 0, but this is unknown.

The derived feature vector captures numerically the existence (or lack) of shared identity information in the corresponding pattern-pair. Let *V* denote the set of all the feature vectors derived from *G*. The derived set of feature vectors corresponding to the previous example (see Figure 7) is as follows:

$$V = \left(v_{01} = \begin{vmatrix} ? \\ 1 \\ ? \\ ? \\ 0 \end{vmatrix}, v_{02} = \begin{vmatrix} 1 \\ ? \\ 1 \\ ? \\ ? \end{vmatrix}, v_{03} = \begin{vmatrix} 0 \\ ? \\ ? \\ 0 \\ ? \end{vmatrix}, v_{12} = \begin{vmatrix} ? \\ ? \\ ? \\ ? \\ ? \end{vmatrix}, v_{13} = \begin{vmatrix} ? \\ ? \\ ? \\ 1 \\ ? \end{vmatrix}, v_{23} = \begin{vmatrix} 0 \\ ? \\ ? \\ ? \\ ? \end{vmatrix} \right)$$

5.2 Fraud detection

We present, in this section, our fraud detection approach, followed by an overview of the RBDT-1 method, and the initial rule base.

5.2.1 Approach

Our fraud detection approach consists of analyzing the coherence or consistency of the shared identity information between patterns. For instance, two identity certificates attributed to the same individual are expected to bear the same birth date and mother maiden name, when such information is available. The feature vectors describing patterns connections are used as basis for such analysis. As an outcome of the analysis, patterns connections are classified in one of four categories: *normal*, *suspicious-low*, *suspicious-high*, or *fraudulent* represented by the labels *N*, *S-*, *S+*, and *F*, respectively. A normal connection is a connection for which no anomaly has been found. We consider two strains or levels of suspicion, *suspicious-low* and *suspicious-high* that refer to unlikely and highly unlikely situations, respectively. A suspicious connection may potentially involve some fraud; the judgment is not definitive, and as such it calls for further inspection. In contrast, a fraudulent connection is a profile involving definitely some fraud; the judgment here is final.

Our fraud detector is implemented as a decision tree that takes pattern-pair feature vectors as an input, and provides as an output a fraud decision. The tree is built and updated dynamically based on a rule base which encodes expert knowledge or common sense understanding of the notion of fraudulent or inconsistent behaviour. No previous fraud instances should be required. The size of the decision tree resulting from such process is expected to be manageable because typically application forms involve sparse and limited identity attributes.

We use a new rule-based decision tree method named RBDT-1 [5, 6] to transform the set of rules into a decision tree. In the rest of this subsection, we give an overview of the RBDT-1 method, and then summarize and discuss a sample of the rules.

5.2.2 The RBDT-1 method

RBDT-1 is a new rule based decision tree method for learning a decision tree from a set of decision rules that cover some data instances. *RBDT-1* method uses a set of declarative rules as an input for generating a decision tree. The method's goal is to create on-demand a short and accurate decision tree from a stable or dynamically changing set of rules. The rules used by *RBDT-1* could be generated

either by an expert, or by an inductive rule learning program. There is a major difference between building a decision tree from examples and building it from rules. When building a decision tree from rules the method assigns attributes to the nodes using criteria based on the properties of the attributes in the decision rules, rather than statistics regarding their coverage of the data examples. To derive the tree, the *RBDT-1* method uses in sequence three different criteria to determine the fit (best) attribute for each node of the tree, which are referred to as *the attribute effectiveness (AE)*, *the attribute autonomy (AA)*, and *the minimum value distribution (MVD)*.

- *Attribute Effectiveness*: the first criterion to be examined for the attributes prefers an attribute which has the most influence in determining the decision classes. In other words, it prefers the attribute that has the least number of "don't care" values for the class decisions in the rules, as this indicates its high relevance for discriminating among rule-sets of given decision classes.
- *Attribute Autonomy*: the second criterion to be examined for the attributes in the *RBDT-1 method*. This criterion is examined when the highest *AE* score is obtained by more than one attribute. This criterion prefers the attribute that will decrease the number of subsequent nodes required ahead in the branch before reaching a leaf node. Thus, it selects the attribute that is less dependent on the other attributes in deciding on the decision classes.
- *Minimum Value Distribution (MVD)*: is concerned with the number of values that an attribute has in the current rules. When the highest *AA* score is obtained by more than one attribute, this criterion selects the attribute with the minimum number of values in the current rules. This criterion minimizes the size of the tree because the fewer the number of values the fewer the number of branches involved and consequently the smaller the tree will be.

In the decision tree building process, we select the fit attribute that will be assigned to each node from the current set of rules *CR* based on the attribute selection criteria outlined above. *CR* is a subset of the decision rules that satisfy the combination of attribute values assigned to the path from the root to the current node. *CR* will correspond to the whole set of rules at the root node. From each node a number of branches are pulled out according to the total number of values available for the corresponding attribute in *CR*. Each branch is associated with a reduced set of rules *RR* which is a subset of *CR* satisfying the value of the corresponding attribute.

If *RR* is empty, then a single node will be returned with the value of the most frequent class found in the whole set of rules. Otherwise, if all the rules in *RR* assigned to the branch belong to the same decision class, a leaf node will be created and assigned a value of that decision class. The process continues until each branch from the root node is terminated with a leaf node and no more further branching is required.

5.2.3 Rule base

Based on the five attributes patterns considered in our previous example, we derived 82 rules for the initial implementation of our proof concept. Table 2 illustrates a decision table for a sample of the rules. In the rules "yes" corresponds to $\delta_{ijl} = 1$, "no" corresponds to $\delta_{ijl} = 0$, "na" corresponds to $\delta_{ijl} = ?$ and * is a placeholder for all possible values of δ_{ijl} (i.e. 1, 0, "?").

Table 2: Sample of the rules in the rule-base

Attributes / Rules #	ssn	mmn	dob	add	tel	decicison
1	no	yes	yes	na	na	s-
2	no	yes	yes	na	no	s-
3	na	no	yes	yes	*	s-
4	no	*	yes	yes	*	s+
5	yes	*	no	*	*	f
6	yes	no	yes	*	*	f
7	yes	no	na	*	*	f
8	*	*	*	no	yes	f
9	no	*	*	na	yes	f
10	no	no	*	no	no	n
11	na	no	*	no	no	n
12	no	na	*	no	no	n
13	na	na	*	no	no	n
14	no	no	*	na	no	n
15	yes	yes	yes	yes	yes	n

While some of the rules are straightforward, many require some explanations. A straightforward fraud case is when the social security numbers match and either the dates of birth or the mother maiden names do not such as rules #5, #6 and #7 in Table 2. Some other straightforward cases of fraud are when the home telephone numbers match while the addresses do not such as rule #8.

In both cases one could assume that the same individual is impersonating two different individuals: in the first case by changing either the mother maiden name or the date of birth, and in the second case by using different locations.

Two straightforward normal cases are when either none of the attributes match or all the attributes match such as rules #10 and #15, respectively. We assume in the case when none of the attributes matches that the non-matching patterns belong to different individuals and have not been used by the applicant (in some past attempt to defraud the system). Obviously, the case when all the attributes match is regular. A variant of this case is when everything else match except the telephone numbers; this is considered normal because it could simply be the case that the same individual owns several telephone lines.

According to the degree of rarity the case is classified as either highly suspicious or lightly suspicious. For instance, considering that we are dealing with homonyms, a case where the social security numbers do not match, while the addresses, dates of birth, and mother maiden names match, such as rule #4, is extremely unusual. Because this simply means that there are two different individuals (different ssn), who are homonyms, born the same day, from homonym mothers and living at the same location. This is highly

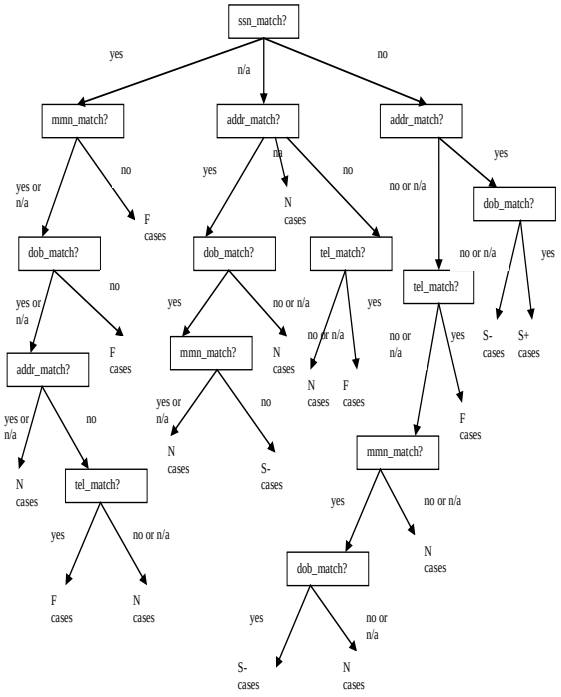
suspicious and requires further verification. If these two individuals (born on the same day with the same mother maiden name) were living in different locations, with different phone numbers, then the situation could be considered as a coincidence, although still uncommon. We classify such occurrence as lightly suspicious. Further investigation would also be required in this case as well, but such investigation does not have to be as thorough as in a highly suspicious case.

The decision tree obtained, by applying the RBDT-1 method to the rule base, is illustrated in Figure 8. The resulting tree consists of 57 rules; the distribution of the different decisions is summarized in Table 3.

Table 3: Rules distribution summary for the RBDT-1 decision tree

Decision	#Rules
Normal (N)	39
Fraud (F)	10
Suspicious-Low (S-)	7
Suspicious-High (S+)	1
Total	57

Figure 8. The decision tree produced by RBDT-1.



6. EVALUATION

We present, in this section, the evaluation of the different components of our fraud detection framework. We start by presenting the evaluation method and data collection, and then we present and discuss the evaluation results obtained for the identity information retrieval engine, followed by the fraud detector.

6.1 Evaluation method and data

The most common way of evaluating frameworks like the one proposed in this paper consists of finding and using some public dataset containing real identity fraud information. As expected, however, the evaluation of our framework faces difficulty in obtaining real data to effectively analyze and test our system. Usually data containing real identity (fraud) information is restricted from being used due to privacy issues. An alternative approach may consist of generating synthetic fraud data. In this work, we collected real identity information online and then generated and injected synthetic fraud information in the data. Having, however, a dataset is not enough to carry out the evaluation. We need an unbiased mechanism to label the data. In the absence of a domain expert to label the data, we have decided to adopt a comparative labelling mechanism in this work. More specifically, we used the communal scoring approach by Phua *et al.* described above in the related work to produce an initial set of labels, and then we compared the produced labels to the labels obtained while using our proposed approach. Using such comparative evaluation allows us to assess the relative strength of our approach.

To obtain real identity information, we conducted some white hat Google hacking experiments over two weeks. During this study, we searched manually for identity information using Google. We used different words that express identity information such as the social security number and searched in different types of documents, and were able to collect sensitive identity information for living as well as for dead persons. The study allowed us to establish a database of online identity information related to 154 different individuals, which serve as basis for the validation of our identity fraud detection framework. Details of the study and corresponding results can be found in [14].

6.2 Evaluation of the information retrieval engine

In order to evaluate our search algorithms, described earlier, firstly, we used Google search engine to manually search for pages containing identity information for only 70 different names out of the 154 names identified in our initial manual search described in the previous section. Those 70 names were the subset of the 154 names for which identity information were still available online at the time of this evaluation; the information corresponding to the remaining names were removed between the time of the initial data collection and the online evaluation reported here. In our search query we used the search identity keywords listed in Table 1 along with each of the 70 names.

The search results for each name were classified into two categories:

- *Keyword-only documents*, which are documents that contain some keywords but no values for any of them.
- *Keyword-value documents*, which are documents that contain at least one value corresponding to an identity keyword.

Secondly, we ran our identity information retrieval tool for each of the 70 names used in the previous manual search. The results of each run consist of document names along with a list of identity keywords and values displayed only for those documents that contain at least one value corresponding to an identity keyword. The numbers of such documents, which are categorized as Keyword_Value documents, as well as the execution time that each search takes are also displayed in the output.

These experiments were conducted on a COMPAQ PC (2 Ghz, 512 Mb of RAM) with AMD Sempron 3000+ and the programs were written using PYTHON language.

We used three metrics to assess the performance of our automated information retrieval scheme, namely recall, precision and false alarm. In the context of our identity information retrieval scheme, these metrics can be defined as follows:

Recall: is the proportion of the correct Keyword_Value documents retrieved by the application from the set of total Keyword_Value documents calculated in the manual search. Recall can be calculated as in (2):

$$Recall = \frac{Proportion\ of\ Correct\ Keyword\_Value\ documents\ calculated\ by\ the\ tool(CKVT)}{Total\ number\ of\ Keyword\_Value\ documents\ calculated\ manually(KVM)} \quad (2)$$

Precision: is the proportion of the correct Keyword_Value documents retrieved by the application from the set of total Keyword_Value documents claimed in the application output. Precision can be calculated as in (3):

$$Precision = \frac{Proportion\ of\ Correct\ Keyword\_Value\ documents\ calculated\ by\ the\ tool(CKVT)}{Total\ number\ of\ Keyword\_Value\ documents\ calculated\ by\ the\ tool(KVT)} \quad (3)$$

False Alarm: is the proportion of the false Keyword_Value documents retrieved by the Identity Information Detector (IID) from the set of total Keyword_Only documents calculated in the manual search. False alarm can be computed as in (4):

$$False\ Alarm = \frac{\text{Proportion of False Keyword\_Value documents calculated by the Tool (FKVT)}}{\text{Total number of keyword\_Only documents calculated Manually (KOM)}} \quad (4)$$

We calculated the recall, precision, and false alarm rates for each of the 70 names, based on corresponding search results. Overall we obtained an average recall of 97.16%, an average precision of 81.82 % and an average false alarm of 13.33%. In general these figures are acceptable since we are collecting information from documents with different formats and structures. For instance, comparable results were obtained in [11] and [12] although the information in their case was collected only from one source of documents which are curriculum vitas.

6.3 Evaluation of the fraud detector

We present, in this section, the application fraud data used to validate our fraud detector and the technique used to label the data. Finally we present the results produced by our fraud detector based on the labelled data.

6.3.1 Application fraud data

In order to evaluate our application fraud detector, we need instances of labelled data both with normal, fraud and suspicious cases and then we need to label the data with our proposed fraud detector and compare both labels.

As mentioned above, due to privacy and confidentiality reasons, obtaining real identity fraud information is extremely difficult. Usually the available real data is encrypted and key identity attributes are removed which makes it ineffective for evaluating our detector.

To overcome this issue we assume that the data that we collected in our white hat Google hacking experiments are application forms submitted by applicants applying for an identity certificate, which yields 154 normal applications. In addition we used the names of each of the 154 applicants to search for identity patterns containing identity information that share the same name.

Since the data that we collected does not include fraud instances, we created some synthetic fraud data based on obvious fraud concepts. Specifically, the main obvious fraud concept used in our evaluation was based on the case where pair of applications bear the same social security number and have either different dates of birth or different mother maiden names. Based on this fraud concept, we created synthetic data based on all the possible value combinations of the identity attributes involved. The synthetic data produced is $(2\times 3^3) - 9$(redundant cases) = 45 fraud data instances. So, overall our evaluation dataset involved 199 data instances consisting of 154 normal cases and 45 fraud cases.

6.3.2 Data labeling

We used a methodology based on a technique proposed by Phua *et al.* in [8] for labelling the data instances described in the previous section as normal, fraud or suspicious.

In order to make the paper self-contained, we briefly describe Phua *et al.* technique which we will refer to as CASS for communal analysis suspicion scoring. CASS is a technique for generating numeric suspicion scores for credit applications based on implicit links to other previous applications over both time and space.

Every new application v_i is pair-wise matched against previous scored applications within a window W.

The attribute vector y_{ij} between v_i and v_j which captures the relationship for an application-pair is defined as:

$$y_{ij} = \{y_1^{ij}, y_2^{ij}, ..., y_N^{ij}\}, \forall i, j$$

Where N is the total number of attributes and y_k^{ij} is the individual match score of each pair-wise attribute: $y_k^{ij} = y_k\left[a_{ik}, a_{jk}\right]$, $1 \le k \le N$.

The match score of each pair-wise attribute y_k could be Boolean 1 for a match or 0 for a non match. In this case the maximum possible score of an application–pair is $\sum_{k=1}^{N} y_k^{ij} = N$. By assigning weights to the attributes, the maximum possible score of an application–pair would be $\sum_{k=1}^{N} y_k^{ij} = 1$.

With their technique, Phua *et al.* label a new application v_i by first linking it with an existing application v_j from either a black list or a white list or as anomalous, or by considering it unlinked. Linkage between v_i and v_j is denoted as $v_i \xrightarrow{L} v_j$, where L is a label, corresponding to *fraud*, or *normal* or *anomalous*. After establishing the linkage, the communal suspicion score $W_{communal_{ij}}$ for the linked application-pair is computed accordingly.

Phua *et al.* assume that the black list is made up of actual identity applications previously found to be fraudulent. As a result, any new application that contains similar identity information to those applications in the black list is considered as a fraud.

For the black list, Phua *et al.* define the communal link derived from pair-wise matching of attributes $W_{communal_{ij}}$ as in (5):

$$W_{communal_{ij}} = \begin{cases} 1 \text{ and } v_i \xrightarrow{fraud} v_j \text{ if } v_j \text{ is a known fraud} \\ \qquad\qquad \text{and } \sum_{k=1}^{N} S(y_k^{ij}) \geq T_{fraud} \\ 0 \quad \text{otherwise} \end{cases} \quad (5)$$

Where T_{fraud} is an integer threshold for linking a new application to the black list ($0 \leq T_{fraud} \leq N$) and S is a string similarity metric, which can be exact matching or fuzzy matching. We use in our evaluation exact matching.

Although the white list is the opposite of the black list, it is not made up necessarily of the actual identity applications that are not fraudulent. The white list consists of a set R of N relationships (or rules) defined as normal: $R = [R_1, R_2, R_3, ..., R_N]$. A weight WR_k is associated with each relationship R_k $(1 \leq k \leq N)$, where $0.5 \leq WR_k \leq 1$. The set of weights $W_{normal} = [WR_1, WR_2, ..., WR_N]$ is sorted in ascending order while the set of relationships R is ranked in descending order of WR_k.

For the white list, the communal score $W_{communal}$ is defined as in (6):

$$W_{communal_{ij}} = \begin{cases} [WR_x * \sum_{k=1}^{N} S(y_k^{ij})] \text{ and } v_i \xrightarrow{normal} v_j \text{ if } y_{ij} \in R \\ \qquad\qquad \text{and } \sum_{k=1}^{N} S(y_k^{ij}) \geq T_{normal} \\ 0 \quad \text{otherwise} \end{cases} \quad (6)$$

Where T_{normal} is an integer threshold for linking a new application to the white list ($0 \leq T_{normal} \leq N$), and WR_x is the weight associated with relationship $R_x \in R$ matching with y_{ij}.

Phua et al. assume that anomalous application pairs correspond to linked applications that are in neither the black list nor the white list. The communal score $W_{communal}$ is defined in this case as in (7):

$$W_{communal_{ij}} = \begin{cases} \sum_{k=1}^{N} S(y_k^{ij}) \text{ and } v_i \xrightarrow{anomalous} v_j \text{ if } y \notin R \\ \qquad\qquad \text{and } \sum_{k=1}^{N} S(y_k^{ij}) \geq T_{normal} \\ 0 \quad \text{otherwise} \end{cases} \quad (7)$$

Unlinked application-pairs are the results of too few attributes (below the selected thresholds) or no attribute match at all. Phua *et al.* state that there is a strong possibility that many fraudulent applications that share few (below the fraud threshold) identity attributes or no attribute with those in the black list would not be detected and would be accepted as normal.

In order to make a decision for a new application, the total suspicion score for that application is calculated based on all the linked application pairs' scores, and then a final classification as fraud, normal or anomalous is made [8].

In the process of labeling our data using Phua *et al.*'s technique, we populated the black list with identity information of four synthetic fraud applications crafted based on the obvious fraud case explained in the previous section. These will be used to match against our 45 synthetic fraud applications mentioned in the previous section. We populated the white list with 15 normal relationships defined by Phua *et al.* in [15] after adapting them to fit the five attributes used in our fraud detector. We randomly selected 49 patterns from the identity patterns that we collected online and labeled them as normal to link them to the white list.

We assumed that the suspicion scores of the applications linked to the black list were very high, and those linked to the white list were very low. Thus, in this case we labeled a new application linked to an application in the black list with $W_{communal} = 1$ as a fraud application. We labeled the applications linked to the white list with $W_{communal} > 0$ and the unlinked applications as normal applications and labeled those linked as anomalous with $W_{communal} > 0$ as suspicious.

6.3.3 Evaluation

To evaluate our application fraud detector we removed the labels of the data and allowed our system to produce its own labels, and then compared them to the data labels produced by CASS. The outcome of the comparison is the *Match Rate (MR)*, when the labels produced by both techniques coincide. The *Non Match Rate (NMR)*, which is the complement of the MR (NMR=1-MR) measures the disagreement between both techniques. Note that a high NMR doesn't necessarily mean a weakness of either technique. A closer error analysis needs to be done to find out which technique is actually at fault.

We chose the values 2 and 3 for the normal and fraud thresholds, which are considered intermediate values since the application-pair matching score is based on five attributes. As a result of setting the thresholds for fraud and normal to a combination of the above two values (2 and 3) we produced four different sets of labels for our data. We compared the labels produced by our proposed fraud detector to each of the four sets of labels and computed the match rate by our method which is summarized in Table 4.

Table 4: Summary of the match rates produced by our fraud detector for different thresholds values when compared to CASS-labeled data

T_{fraud}	T_{normal}	Match Rate
2	2	87%
3	3	82%
3	2	77%
2	3	92%

Based on the selected thresholds, the best match rate produced by our proposed fraud detector was 92% for the data labelled by CASS using a combination of $T_{normal} = 3$ for the white list and the anomalous and $T_{fraud} = 2$ for the black list.

To analyze this result, we have to discuss the labels produced by the CASS approach. By reviewing the 45 synthetic fraud applications that we created, we discovered that 12 fraud applications were labelled as unlinked by CASS, and as a result were considered as normal and got accepted. These 12

applications are cases where two applications share only one attribute value (below the black list threshold) that happens to be the social security number. Although these applications share only one attribute, they correspond to an obvious fraud case because the social security number is a unique attribute and hence requires the rest of the attributes to be identical. Thus, in that sense our application fraud detector produces a correct fraud label for all the 45 fraud applications which, however, does not match the CASS labels. CASS produces the wrong labels because, as explained above, the method depends on a threshold which is not effective in detecting such fraud cases.

For the 154 normal applications, while our application fraud detector labels them as normal, CASS labelled 3 of the 154 applications as anomalous and the rest as normal. Examination of these 3 CASS-labelled anomalous applications shows that while the threshold for a linkage between each of the applications and a normal application is met there is no corresponding matching relationship in R. This could possibly be addressed by expanding the initial set of normal relationships used for the evaluation.

Overall our proposed system performs well when assessed against data labelled using CASS, which underscores its potential as a strong fraud detector.

7. CONCLUSION

In the last several years, identity theft has been on the rise. Unfortunately, the Internet has been facilitating this phenomenon since it represents a tremendous and open repository for sensitive identity information available for those who know how to find them, including fraudsters. We have presented in this paper an unsupervised application fraud detection framework that analyzes online identity patterns using an intelligent decision model to identify fraudulent applications. We designed a web-based identity information retrieval engine capable of finding and collecting online identity information. We defined heuristic rules for detecting fraudulent applications and used a rule-based decision tree method for transforming the rules into a decision tree. To evaluate our fraud detector we used a mix of real data collected online and synthetic data to induce some fraud cases. The data was treated as identity applications and labelled using a technique called CASS as normal, fraud or suspicious. The data was labelled four times based on the CASS approach by using different combinations of normal and fraud thresholds and compared to the labels produced by our fraud detector. The best match rate produced by our detector was 92%. The non-matching cases happened to be due to labelling errors by the CASS approach. Overall our approach shows strong promise for online identity application fraud detection.

In future work, we intend to extend and apply our fraud detection approach to other possibly more traditional data sources.

References

[1] Crown "Identity fraud: A study", Economic and Domestic Secretariat Cabinet Office, www.homeoffice.gov.uk/docs/id_fraud-report.PDF, 2002.

[2] W. Wang; Y. Yuan; N. Archer. "A Contextual Framework for Combating Identity Theft", Security & Privacy Magazine, IEEE Volume 4, pp. 30 – 38, 2006.

[3] Bolton, R. J. and Hand, D. J. "Unsupervised profiling methods for fraud detection", Credit Scoring and Credit Control, Vol. 2, pp.5–7, 2001.

[4] I. F. Imam, and R. S. Michalski "Should Decision Trees be Learned from Examples of From Decision Rules?", Source Lecture Notes in Computer Science. In Proceedings of the 7th International Symposium on Methodologies, 689, pp. 395–404, 1993.

[5] A. Abdelhalim, I.Traore.'Converting Declarative Rules into Decision Trees', International Conference on Computer Science and Applications, San Francisco, USA, 20-22 October, 2009.

[6] A. Abdelhalim, I.Traore, B. Sayed. 'RBDT-1: a New Rule-based Decision Tree Generation Technique', 3rd International Symposium on Rules, Applications and Interoperability, Las Vegas, Nevada, USA: Nov 5-7, 2009.

[7] P. Burns, A. Stanley. "Fraud Management in the Credit Card Industry", Payment Cards Center Discussion Paper number 02-05, Federal Reserve Bank of Philadelphia.http://www.phil.frb.org/pcc/papers/2002/FraudManagement_042002.pdf, 2002.

[8] R. Wheeler, S. Aitken. "Multiple Algorithms for Fraud Detection", Knowledge-Based Systems, Vol. 13, No. 3, pp.93–99, 2000.

[9] C. Phua, V. Lee, K. Smith, R. Gayler, "A Comprehensive Survey of Data Mining-Based Fraud Detection Research", *Artificial Intelligence Review, submitted*, 2005. Available at: http://www.bsys.monash.edu.au/people/cphua/.

[10] P. A. Estevez, C. M. Held, C. A. Perez, "Subscription Fraud Prevention in Telecommunications Using Fuzzy Rules And Neural Networks", Expert Systems with Applications, Vol. 31, pp. 337–344, 2006.

[11] L. Sweeney. "AI Technologies to Defeat Identity Theft Vulnerabilities", AAAI Spring Symposium on AI Technologies for Homeland Security, 2005.

[12] L. Sweeney. "Protecting Job Seekers from Identity Theft", IEEE Internet Computing, Vol. 10, No. 2, pp.74–78, 2006.

[13] Chan, P. K., Fan, W., Prodromidis, A. L. and Stolfo, S. J. "Distributed Data Mining in Credit Card Fraud Detection", Intelligent Systems, IEEE, Vol. 14, No. 6, pp.67–74, 1999.

[14] Abdelhalim, A. and Traore, I. "The Impact of Google Hacking on Identity and Application Fraud", Proc. IEEE Pacific Rim Conference, Communications, Computers and Signal Processing, pp. 240–244, 2007.

[15] Phua, C., Gayler, R., Lee, V. and Smith-Miles, K. "On the Communal Analysis Suspicion Scoring for Identity Crime in Streaming Credit Applications", European Journal of Operational Research, 195, pp. 595-612, 2009.

A. Abdelhalim received an information system analyst degree from University of Helwan in 2000. She held a demonstrator position in information systems department from (2000- 2003). She received her MS.c in Computer Science from University of Helwan in 2003. She worked as an Assistant lecturer at the Department of Information Systems from (2003-2004), Helwan University, Cairo,

Egypt. She is currently a PhD student in Electrical Computer Engineering Department at University of Victoria, Canada.

Issa Traore received an Aircraft Engineer Degree from Ecole de l'Air in Salon de Provence (France) in 1990, and successively two Master Degrees in Aeronautics and Space Techniques in 1994, and in Automatics and Computer Engineering in 1995 from *Ecole Nationale Superieure de l'Aeronautique et de l'Espace* (E.N.S.A.E), Toulouse, France. In 1998, Traore received a PhD in Software Engineering from Institute Nationale Polytechnique (INPT)-LAAS/CNRS, Toulouse, France. From June – October 1998, he held a post-doc position at LAAS-CNRS, Toulouse, France, and Research Associate (November 1998 – May 1999), and Senior Lecturer (June – October 1999) at the University of Oslo. Since November 1999, he has joined the Faculty of the Department of ECE, University of Victoria, Canada. He is currently an Associate Professor. His research interests include behavioral biometrics systems, intrusion detection systems, software security metrics, and software quality engineering. He is the founder and coordinator of the Information Security and Object Technology (ISOT) Lab (http://www.isot.ece.uvic.ca