Resilient Identity Crime Detection

Clifton Phua, *Member*, *IEEE*, Kate Smith-Miles, *Senior Member*, *IEEE*, Vincent Cheng-Siong Lee, and Ross Gayler

Abstract—Identity crime is well known, prevalent, and costly; and credit application fraud is a specific case of identity crime. The existing nondata mining detection system of business rules and scorecards, and known fraud matching have limitations. To address these limitations and combat identity crime in real time, this paper proposes a new multilayered detection system complemented with two additional layers: communal detection (CD) and spike detection (SD). CD finds real social relationships to reduce the suspicion score, and is tamper resistant to synthetic social relationships. It is the whitelist-oriented approach on a fixed set of attributes. SD finds spikes in duplicates to increase the suspicion score, and is probe-resistant for attributes. It is the attribute-oriented approach on a variable-size set of attributes. Together, CD and SD can detect more types of attacks, better account for changing legal behavior, and remove the redundant attributes. Experiments were carried out on CD and SD with several million real credit applications. Results on the data support the hypothesis that successful credit application fraud patterns are sudden and exhibit sharp spikes in duplicates. Although this research is specific to credit application fraud detection, the concept of resilience, together with adaptivity and quality data discussed in the paper, are general to the design, implementation, and evaluation of all detection systems.

Index Terms—Data mining-based fraud detection, security, data stream mining, anomaly detection.

1 Introduction

Identity crime is defined as broadly as possible in this paper. At one extreme, synthetic identity fraud refers to the use of plausible but fictitious identities. These are effortless to create but more difficult to apply successfully. At the other extreme, real identity theft refers to illegal use of innocent people's complete identity details. These can be harder to obtain (although large volumes of some identity data are widely available) but easier to successfully apply. In reality, identity crime can be committed with a mix of both synthetic and real identity details.

Identity crime has become prominent because there is so much real identity data available on the Web, and confidential data accessible through unsecured mailboxes. It has also become easy for perpetrators to hide their true identities. This can happen in a myriad of insurance, credit, and telecommunications fraud, as well as other more serious crimes. In addition to this, identity crime is prevalent and costly in developed countries that do not have nationally registered identity numbers.

Data breaches which involve lost or stolen consumers' identity information can lead to other frauds such as tax returns, home equity, and payment card fraud. Consumers can incur thousands of dollars in out-of-pocket expenses. The US law requires offending organizations to notify consumers, so that consumers can mitigate the harm. As a result, these organizations incur economic damage, such as notification costs, fines, and lost business [24].

Credit applications are Internet or paper-based forms with written requests by potential customers for credit cards, mortgage loans, and personal loans. Credit application fraud is a specific case of identity crime, involving synthetic identity fraud and real identity theft.

As in identity crime, credit application fraud has reached a critical mass of fraudsters who are highly experienced, organized, and sophisticated [10]. Their visible patterns can be different to each other and constantly change. They are persistent, due to the high financial rewards, and the risk and effort involved are minimal. Based on anecdotal observations of experienced credit application investigators, fraudsters can use software automation to manipulate particular values within an application and increase frequency of successful values.

Duplicates (or matches) refer to applications which share common values. There are two types of duplicates: exact (or identical) duplicates have the all same values; near (or approximate) duplicates have some same values (or characters), some similar values with slightly altered spellings, or both. This paper argues that each successful credit application fraud pattern is represented by a sudden and sharp spike in duplicates within a short time, relative to the established baseline level.

Duplicates are hard to avoid from fraudsters' point-of-view because duplicates increase their' success rate. The synthetic identity fraudster has low success rate, and is likely to reuse fictitious identities which have been successful before. The identity thief has limited time because innocent people can discover the fraud early and take action, and will quickly use the same real identities at different places.

- *C. Phua is with the Data Mining Department, Institute for Infocomm Research (I^2R), 1 Fusionopolis Way, #21-01 Connexis (South Tower), Singapore 138632. E-mail: clifton.phua@gmail.com.*
- *K. Smith-Miles is with the School of Mathematical Sciences, Monash University, Wellington Road, Clayton, Victoria 3800, Australia. E-mail: kate.smith-miles@sci.monash.edu.au.*
- *V.C.-S. Lee is with the Clayton School of Information Technology, Monash University, Office 122, Building 63, Wellington Road, Clayton, Victoria 3800, Australia. E-mail: vincent.cs.lee@monash.edu.*
- *R. Gayler is with the Veda Advantage, Level 12, 628 Bourke Street, Melbourne, Victoria 3000, Australia. E-mail: r.gayler@gmail.com.*

Manuscript received 27 Oct. 2009; revised 1 Sept. 2010; accepted 30 Oct. 2010; online 21 Dec. 2010.
Recommended for acceptance by E. Bertino.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2009-10-0739.
Digital Object Identifier no. 10.1109/TKDE.2010.262.

It will be shown later in this paper that many fraudsters operate this way with these applications and that their characteristic pattern of behavior can be detected by the methods reported. In short, the new methods are based on white-listing and detecting spikes of similar applications. White-listing uses real social relationships on a fixed set of attributes. This reduces false positives by lowering some suspicion scores. Detecting spikes in duplicates, on a variable set of attributes, increases true positives by adjusting suspicion scores appropriately.

Throughout this paper, data mining is defined as the real-time search for patterns in a principled (or systematic) fashion. These patterns can be highly indicative of early symptoms in identity crime, especially synthetic identity fraud [22].

1.1 Main Challenges for Detection Systems

Resilience is the ability to degrade gracefully when under most real attacks. The basic question asked by all detection systems is whether they can achieve resilience. To do so, the detection system trades off a small degree of efficiency (degrades processing speed) for a much larger degree of effectiveness (improves security by detecting most real attacks). In fact, any form of security involves tradeoffs [26].

The detection system needs "defence-in-depth" with multiple, sequential, and independent layers of defence [25] to cover different types of attacks. These layers are needed to reduce false negatives. In other words, any successful attack has to pass every layer of defence without being detected.

The two greatest challenges for the data mining-based layers of defence are adaptivity and use of quality data. These challenges need to be addressed in order to reduce false positives.

Adaptivity accounts for morphing fraud behavior, as the attempt to observe fraud changes its behavior. But what is not obvious, yet equally important, is the need to also account for changing legal (or legitimate) behavior within a changing environment. In the credit application domain, changing legal behavior is exhibited by communal relationships (such as rising/falling numbers of siblings) and can be caused by external events (such as introduction of organizational marketing campaigns). This means legal behavior can be hard to distinguish from fraud behavior, but it will be shown later in this paper that they are indeed distinguishable from each other.

The detection system needs to exercise caution with applications which reflect communal relationships. It also needs to make allowance for certain external events.

Quality data are highly desirable for data mining and data quality can be improved through the real time removal of data errors (or noise). The detection system has to filter duplicates which have been reentered due to human error or for other reasons. It also needs to ignore redundant attributes which have many missing values, and other issues.

1.2 Existing Identity Crime Detection System

There are nondata mining layers of defence to protect against credit application fraud, each with its unique strengths and weaknesses.

The first existing defence is made up of business rules and scorecards. In Australia, one business rule is the hundred-point physical identity check test which requires the applicant to provide sufficient point-weighted identity documents face-to-face. They must add up to at least 100 points, where a passport is worth 70 points. Another business rule is to contact (or investigate) the applicant over the telephone or Internet. The above two business rules are highly effective, but human resource intensive. To rely less on human resources, a common business rule is to match an application's identity number, address, or phone number against external databases. This is convenient, but the public telephone and address directories, semipublic voters' register, and credit history data can have data quality issues of accuracy, completeness, and timeliness. In addition, scorecards for credit scoring can catch a small percentage of fraud which does not look creditworthy; but it also removes outlier applications which have a higher probability of being fraudulent.

The second existing defence is known fraud matching. Here, known frauds are complete applications which were confirmed to have the intent to defraud and usually periodically recorded into a blacklist. Subsequently, the current applications are matched against the blacklist. This has the benefit and clarity of hindsight because patterns often repeat themselves. However, there are two main problems in using known frauds. First, they are untimely due to long time delays, in days or months, for fraud to reveal itself, and be reported and recorded. This provides a window of opportunity for fraudsters. Second, recording of frauds is highly manual. This means known frauds can be incorrect [11], expensive, difficult to obtain [21], [3], and have the potential of breaching privacy.

In the real-time credit application fraud detection domain, this paper argues against the use of classification (or supervised) algorithms which use class labels. In addition to the problems of using known frauds, these algorithms, such as logistic regression, neural networks, or Support Vector Machines (SVM), cannot achieve scalability or handle the extreme imbalanced class [11] in credit application data streams. As fraud and legal behavior changes frequently, the classifiers will deteriorate rapidly and the supervised classification algorithms will need to be trained on the new data. But the training time is too long for real-time credit application fraud detection because the new training data have too many derived numerical attributes (converted from the original, sparse string attributes) and too few known frauds.

This paper acknowledges that in another domain, real-time credit card *transactional* fraud detection, there are the same issues of scalability, extremely imbalanced classes, and changing behavior. For example, FairIsaac—a company renown for their predictive fraud analytics—has been successfully applying supervised classification algorithms, including neural networks and SVM.

1.3 New Data Mining-Based Layers of Defence

The main objective of this research is to achieve resilience by adding two new, real time, data mining-based layers to complement the two existing nondata mining layers discussed in the section. These new layers will improve

detection of fraudulent applications because the detection system can detect more types of attacks, better account for changing legal behavior, and remove the redundant attributes.

These new layers are not human resource intensive. They represent patterns in a score where the higher the score for an application, the higher the suspicion of fraud (or anomaly). In this way, only the highest scores require human intervention. These two new layers, communal and spike detection, do not use external databases, but only the credit application database *per se*. And crucially, these two layers are unsupervised algorithms which are not completely dependent on known frauds but use them only for evaluation.

The main contribution of this paper is the demonstration of resilience, with adaptivity and quality data in real-time data mining-based detection algorithms. The first new layer is Communal Detection (CD): the whitelist-oriented approach on a fixed set of attributes. To complement and strengthen CD, the second new layer is Spike Detection (SD): the attribute-oriented approach on a variable-size set of attributes.

The second contribution is the significant extension of knowledge in credit application fraud detection because publications in this area are rare. In addition, this research uses the key ideas from other related domains to design the credit application fraud detection algorithms.

Finally, the last contribution is the recommendation of credit application fraud detection as one of the many solutions to identity crime. Being at the first stage of the credit life cycle, credit application fraud detection also prevents some credit *transactional* fraud.

Section 2 gives an overview of related work in credit application fraud detection and other domains. Section 3 presents the justifications and anatomy of the CD algorithm, followed by the SD algorithm. Before the analysis and interpretation of CD and SD results, Section 4 considers the legal and ethical responsibility of handling application data, and describes the data, evaluation measures, and experimental design. Section 5 concludes the paper.

2 Background

Many individual data mining algorithms have been designed, implemented, and evaluated in fraud detection. Yet until now, to the best of the researchers' knowledge, resilience of data mining algorithms in a complete detection system has not been explicitly addressed.

Much work in credit application fraud detection remains proprietary and exact performance figures unpublished, therefore there is no way to compare the CD and SD algorithms against their leading industry methods and techniques. For example, [14] has *ID Score-Risk* which gives a combined view of each credit application's characteristics and their similarity to other industry-provided or Web identity's characteristics. In another example, [7] has *Detect* which provides four categories of policy rules to signal fraud, one of which is checking a new credit application against historical application data for consistency.

Case-based reasoning (CBR) is the only known prior publication in the screening of credit applications [29]. CBR analyzes the hardest cases which have been misclassified by existing methods and techniques. Retrieval uses thresholded nearest neighbor matching. Diagnosis utilizes multiple selection criteria (probabilistic curve, best match, negative selection, density selection, and default) and resolution strategies (sequential resolution-default, best guess, and combined confidence) to analyze the retrieved cases. CBR has 20 percent higher true positive and true negative rates than common algorithms on credit applications.

The CD and SD algorithms, which monitor the significant increase or decrease in amount of something important (Section 3), are similar in concept to credit *transactional* fraud detection and bioterrorism detection. Peer group analysis [2] monitors interaccount behavior over time. It compares the cumulative mean weekly amount between a target account and other similar accounts (peer group) at subsequent time points. The suspicion score is a t-statistic which determines the standardized distance from the centroid of the peer group. On credit card accounts, the time window to calculate a peer group is 13 weeks, and the future time window is 4 weeks. Break point analysis [2] monitors intraaccount behavior over time. It detects rapid spending or sharp increases in weekly spending within a single account. Accounts are ranked by the t-test. The fixed-length moving transaction window contains 24 transactions: the first 20 for training and the next four for evaluation on credit card accounts. Bayesian networks [31] uncover simulated anthrax attacks from real emergency department data. Wong [30] surveys algorithms for finding suspicious activity in time for disease outbreaks. Goldenberg et al. [9] use time series analysis to track early symptoms of synthetic anthrax outbreaks from daily sales of retail medication (throat, cough, and nasal) and some grocery items (facial tissues, orange juice, and soup). Control-chart-based statistics, exponential weighted moving averages, and generalized linear models were tested on the same bioterrorism detection data and alert rate [15].

The SD algorithm, which specifies how much the current prediction is influenced by past observations (Section 3.3), is related to Exponentially Weighted Moving Average (EWMA) in statistical process control research [23]. In particular, like EWMA, the SD algorithm performs linear forecasting on the smoothed time series, and their advantages include low implementation and computational complexity. In addition, the SD algorithm is similar to change point detection in biosurveillance research, which maintains a cumulative sum (CUSUM) of positive deviations from the mean [13]. Like CUSUM, the SD algorithm raises an alert when the score/CUSUM exceeds a threshold, and both detects change points faster as they are sensitive to small shifts from the mean. Unlike CUSUM, the SD algorithm weighs and chooses string attributes, not numerical ones.

3 The Methods

This section is divided into four sections to systematically explain the CD algorithm (first two sections) and the SD algorithm (last two sections). Each section commences with a clearer discussion about its purposes.

3.1 Communal Detection

This section motivates the need for CD and its adaptive approach.

Suppose there were two credit card applications that provided the same postal address, home phone number, and date of birth, but one stated the applicant's name to be Jo*h*n Smith, and the other stated the applicant's name to be Jo*a*n Smith. These applications could be interpreted in three ways:

1. Either it is a fraudster attempting to obtain multiple credit cards using near duplicated data.
2. Possibly there are twins living in the same house who both are applying for a credit card.
3. Or it can be the same person applying twice, and there is a typographical error of one character in the first name.

With the CD layer, any two similar applications could be easily interpreted as (1) because this paper's detection methods use the similarity of the current application to all prior applications (not just known frauds) as the suspicion score. However, for this particular scenario, CD would also recognize these two applications as either (2) or (3) by lowering the suspicion score due to the higher possibility that they are legitimate.

To account for legal behavior and data errors, CD is the whitelist-oriented approach on a fixed set of attributes. The whitelist, a list of communal and self-relationships between applications, is crucial because it reduces the scores of these legal behaviors and false positives. Communal relationships are near duplicates which reflect the social relationships from tight familial bonds to casual acquaintances: family members, housemates, colleagues, neighbors, or friends [17]. The family member relationship can be further broken down into more detailed relationships such as husband-wife, parent-child, brother-sister, male-female cousin (or both male, or both female), as well as uncle-niece (or uncle-nephew, auntie-niece, auntie-nephew). Self-relationships highlight the same applicant as a result of legitimate behavior (for simplicity, self-relationships are regarded as communal relationships).

Broadly speaking, the whitelist is constructed by ranking link-types between applicants by volume. The larger the volume for a link-type, the higher the probability of a communal relationship. On when and how the whitelist is constructed, please refer to Section 3.2, Step 6 of the CD algorithm.

However, there are two problems with the whitelist. First, there can be focused attacks on the whitelist by fraudsters when they submit applications with synthetic communal relationships. Although it is difficult to make definitive statements that fraudsters will attempt this, it is also wrong to assume that this will not happen. The solution proposed in this paper is to make the contents of the whitelist become less predictable. The values of some parameters (different from an application's identity value) are automatically changed such that it also changes the whitelist's link types. In general, tampering is not limited to hardware, but can also refer to manipulating software such as code. For our domain, tamper resistance refers to making it more difficult for fraudsters to manipulate or circumvent data mining by providing false data.

Second, the volume and ranks of the whitelist's real communal relationships change over time. To make the whitelist exercise caution with (or more adaptive to) changing legal behavior, the whitelist is continually being reconstructed.

3.2 CD Algorithm Design

This section explains how the CD algorithm works in real time by giving scores when there are exact or similar matches between categorical data; and in terms of its nine inputs, three outputs, and six steps.

This research focuses on one rapid and continuous data stream [19] of applications. For clarity, let G represent the overall stream which contains multiple and consecutive $\{\ldots, g_{x-2}, g_{x-1}, g_x, g_{x+1}, g_{x+2}, \ldots\}$ Minidiscrete streams.

- g_x: current Minidiscrete stream which contains multiple and consecutive $\{u_{x,1}, u_{x,2}, \ldots, u_{x,p}\}$ microdiscrete streams.
- x: fixed interval of the current month, fortnight, or week in the year.
- p: variable number of microdiscrete streams in a Minidiscrete stream.

Also, let $u_{x,y}$ represent the current microdiscrete stream which contains multiple and consecutive $\{v_{x,y,1}, v_{x,y,2}, \ldots, v_{x,y,q}\}$ applications. The current application's links are restricted to previous applications within a moving window, and this window can be larger than the number of applications within the current microdiscrete stream.

- y: fixed interval of the current day, hour, minute, or second.
- q: variable number of applications in a microdiscrete stream.

Here, it is necessary to describe a single and continuous stream of applications as being made up of separate chunks: a Minidiscrete stream is long-term (for example, a month of applications); while a microdiscrete stream is short-term (for example, a day of applications). They help to specify precisely *when* and *how* the detection system will automatically change its configurations. For example, the CD algorithm reconstructs its whitelist at the end of the month and resets its parameter values at the end of the day; the SD algorithm does attribute selection and updates CD attribute weights at the end of the month. Also, for example, long-term previous average score, long-term previous average links, and average density of each attribute are calculated from data in a Minidiscrete stream; short-term current average score and short-term current average links are calculated from data in a microdiscrete stream.

With this data stream perspective in mind, the CD algorithm matches the current application against a moving window of previous applications. It accounts for attribute weights which reflect the degree of importance in attributes. The CD algorithm matches all links against the whitelist to find communal relationships and reduce their link score. It then calculates the current application's score using every link score and previous application score. At

TABLE 1
Overview of Communal Detection Algorithm

Inputs
v_i (current application)
W number of v_j (moving window)
$\Re_{x,link-type}$ (link-types in current whitelist)
$T_{similarity}$ (string similarity threshold)
$T_{attribute}$ (attribute threshold)
η (exact duplicate filter)
α (exponential smoothing factor)
T_{input} (input size threshold)
SoA (State-of-Alert)

Outputs
$S(v_i)$ (suspicion score)
Same or new parameter value
New whitelist

CD algorithm

Step 1: Multi-attribute link [match v_i against W number of v_j to determine if a single attribute exceeds $T_{similarity}$; and create multi-attribute links if near duplicates' similarity exceeds $T_{attribute}$ or an exact duplicates' time difference exceeds η]

Step 2: Single-link score [calculate single-link score by matching Step 1's multi-attribute links against $\Re_{x,link-type}$]

Step 3: Single-link average previous score [calculate average previous scores from Step 1's linked previous applications]

Step 4: Multiple-links score [calculate $S(v_i)$ based on weighted average (using α) of Step 2's link scores and Step 3's average previous scores]

Step 5: Parameter's value change [determine same or new parameter value through SoA (for example, by comparing input size against T_{input}) at end of $u_{x,y}$]

Step 6: Whitelist change [determine new whitelist at end of g_x]

the end of the current microdiscrete data stream, the CD algorithm determines the SoA and updates one random parameter's value such that it trades off effectiveness with efficiency, or vice versa. At the end of the current Minidiscrete data stream, it constructs the new whitelist.

Table 1 shows the data input, six most influential parameters, and two adaptive parameters.

- v_i: unscored current application. N is its number of attributes. $a_{i,k}$ is the value of the kth attribute in application v_i.
- W: moving (or sliding) window of previous applications. It determines the short time search space for the current application. CD utilizes an application-based window (such as the previous 10,000 applications). v_j is the scored previous application. $a_{j,k}$ is the value of the kth attribute in application v_j.
- $\Re_{x,link-type}$ is a set of unique and sorted link-types (in descending order by number of links), in the link-type attribute of the current whitelist. M is the number of link-types.
- $T_{similarity}$: string similarity threshold between two values.
- $T_{attribute}$: attribute threshold which requires a minimum number of matched attributes to link two applications.
- η: exact duplicate filter at the link level. It removes links of exact duplicates from the same organization within minutes, likely to be data errors by customers or employees.
- α: exponential smoothing factor. In CD, α gradually discounts the effect of average previous scores as the older scores become less relevant.
- T_{input}: input size threshold. When the environment evolves significantly over time, the input size threshold T_{input} may have to be manually adjusted.
- SoA (State-of-Alert): condition of reduced, same, or heightened watchfulness for each parameter.

Table 1 also shows the three outputs.

- $S(v_i)$: CD suspicion score of current application.
- Same or new parameter values for each parameter.
- New whitelist.

While Table 1 gives an overview of the CD algorithm's six steps, the details in each step are presented below.

Step 1: Multiattribute link. The first step of the CD algorithm matches every current application's value against a moving window of previous applications' values to find links

$$e_k = \begin{cases} 1, & \text{if } Jaro-Winkler(a_{i,k}, a_{j,k}) \geq T_{similarity}, \\ 0, & \text{otherwise}, \end{cases} \quad (1)$$

where e_k is the single-attribute match between the current value and a previous value. The first case uses Jaro-Winkler(.) [30], is case sensitive, and can also be cross-matched between current value and previous values from another similar attribute. The second case is a nonmatch because values are not similar

$$e_{i,j} = \begin{cases} e_1 e_2 \ldots e_N, & \text{if } T_{attribute} \leq \sum_{k=1}^{N} e_k \leq N-1 \\ & \quad \text{or } [\sum_{k=1}^{N} e_k = N \\ & \quad \text{and } Time(a_{i,k}, a_{j,k}) \geq \eta], \\ \varepsilon, & \text{otherwise}, \end{cases} \quad (2)$$

where $e_{i,j}$ is the multiattribute link (or binary string) between the current application and a previous application. ε is the empty string. The first case uses Time(.) which is the time difference in minutes. The second case has no link (empty string) because it is not a near duplicate, or it is an exact duplicate within the time filter.

Step 2: Single-link communal detection. The second step of the CD algorithm accounts for attribute weights, and matches every current application's link against the whitelist to find communal relationships and reduce their link score

$$S(e_{i,j}) = \begin{cases} \sum_{k=1}^{N}(e_k \times w_k) \times w_z, & \text{if } e_{i,j} \in \Re_{x,link\text{-}type} \\ & \quad \text{and } e_{i,j} \neq \varepsilon, \\ \sum_{k=1}^{N}(e_k \times w_k), & \text{if } e_{i,j} \notin \Re_{x,link\text{-}type} \\ & \quad \text{and } e_{i,j} \neq \varepsilon, \\ 0, & \text{otherwise}, \end{cases} \quad (3)$$

where $S(e_{i,j})$ is the single-link score. This terminology "single-link score" is adopted over "multiattribute link score" to focus on a single link between two applications, not on the matching of attributes between them. The first case uses w_k which is the attribute weight with default values of $\frac{1}{N}$, and w_z which is the weight of the zth link-type in the whitelist. The second case is the graylist

TABLE 2
Sample of Six Credit Applications with Six Attributes

i or j	Given name	Family name	Unit no.	Street name	Home phone no.	Date of birth
1	John	Smith	1	Circular road	91234567	1/1/1982
2	Joan	Smith	1	Circular road	91234567	1/1/1982
3	Jack	Jones	3	Square drive	93535353	3/2/1955
4	Ella	Jones	3	Square drive	93535353	6/8/1957
5	Riley	Lee	2	Circular road	91235678	5/3/1983
6	Liam	Smyth	2	Circular road	91235678	1/1/1982

(neither in the blacklist nor whitelist) link score. The last case is when there is no multiattribute link.

Step 3: Single-link average previous score. The third step of the CD algorithm is the calculation of every linked previous applications' score for inclusion into the current application's score. The previous scores act as the established baseline level

$$\beta_j = \begin{cases} \dfrac{S(v_j)}{E_O(v_j)}, & \text{if } e_{i,j} \neq \varepsilon \\ & \text{and } E_O(v_j) > 0, \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

where β_j is the single-link average previous score. As there will be no linked applications, the initial values of $\beta_j = 0$ since $S(v_j) = 0$ and $E_o(v_j) = 0$. $S(v_j)$ is the suspicion score of a previous application to which the current application links. $S(v_j)$ was computed the same way as $S(v_i)$—a previous application was once a current application. $E_O(v_j)$ is the number of outlinks from the previous application. The first case gives the average score of each previous application. The last case is when there is no multiattribute link.

Step 4: Multiple-links score. The fourth step of the CD algorithm is the calculation of every current application's score using every link and previous application score

$$S(v_i) = \sum_{v_j \in K(v_i)} [S(e_{i,j}) + \beta_j], \tag{5}$$

where $S(v_i)$ is the CD suspicion score of the current application. $K(v_i)$ is the set of previous applications within the moving window to which the current application links. Therefore, a high score is the result of strong links between current application and the previous applications (represented by $S(e_{i,j})$), the high scores from linked previous applications (represented by β_j), and a large number of linked previous applications (represented by $\sum_{v_j \in K(v_i)}[.]$)

$$S(v_i) = \sum_{v_j \in K(v_i)} [(1-\alpha) \times S(e_{i,j}) + \alpha \times \beta_j], \tag{6}$$

where (6) incorporates α [6] into (5).

Step 5: Parameter's value change. At the end of the current microdiscrete data stream, the adaptive CD algorithm determines the State-of-Alert (SoA) and updates one random parameter's value such that there is a tradeoff between effectiveness with efficiency, or vice versa. This increases the tamper resistance in parameters

$$\text{SoA} = \begin{cases} \text{low}, & \text{if } q \geq T_{input} \text{ and } \Omega_{x-1} \geq \Omega_{x,y}, \\ & \text{and } \delta_{x-1} \geq \delta_{x,y}, \\ \text{high}, & \text{if } q < T_{input} \text{ and } \Omega_{x-1} < \Omega_{x,y}, \\ & \text{and } \delta_{x-1} < \delta_{x,y}, \\ \text{medium}, & \text{otherwise,} \end{cases} \tag{7}$$

where SoA is the state-of-alert at the end of every microdiscrete data stream. Ω_{x-1} is the long-term previous average score and $\Omega_{x,y}$ is the short-term current average score. δ_{x-1} is the long-term previous average links and $\delta_{x,y}$ is the short-term current average links. Collectively, these are termed output suspiciousness.

The first case sets SoA to low when input size is high and output suspiciousness is low. The adaptive CD algorithm trades off one random parameter's effectiveness (degrades communal relationship security) for efficiency (improves computation speed). For example, a smaller moving window, fewer link types in the whitelist, or a larger attribute threshold decreases the algorithm's effectiveness but increases its efficiency.

Conversely, the second case sets SoA to high when its conditions are the opposite of the first case. The adaptive CD algorithm will trade off one random parameter's efficiency (degrades speed) for effectiveness (improves security).

The last case sets SoA to medium. The adaptive CD algorithm will not change any parameter's value.

Step 6: Whitelist change. At the end of the current Minidiscrete data stream, the adaptive CD algorithm constructs the new whitelist on the current Minidiscrete stream's links. This increases the tamper-resistance in the whitelist.

Table 2 provides a sample of six credit applications with six attributes, to show how communal relationships are extracted from credit applications.

The whitelist is constructed from multiattribute links generated from Step 1 of the CD algorithm on the training data. In our simple illustration, the CD algorithm is assumed to have the following parameter settings: $T_{similarity} = 0.8$, $T_{attribute} = 3$, and $M = 4$. If Table 2 is used as training data, five multi-attribute links will be generated: $e_{1,2} = 011111$, $e_{1,6} = 010101$, $e_{2,6} = 010101$, $e_{3,4} = 011110$, and $e_{5,6} = 001110$. These multiattribute links capture communal relationships: John and Joan are twins, Jack and Ella are married, Riley and Liam are housemates, John and Joan are neighbors with Riley and Liam; and John, Joan, and Liam share the same birthday.

Table 3 shows the sample whitelist constructed from credit applications in Table 2. A whitelist contains three attributes. They include the link type, which is a unique link

TABLE 3
Sample Whitelist

z	Link-type	No.	Weight
1	010101	2	0.25
2	011111	1	0.5
3	011110	1	0.75
4	001110	1	1

determined from aggregated links from training data, and its corresponding number of this type of link and its link-type weight. There will be many link types, so the quantity of link types are predetermined by selecting the most frequent ones to be in the whitelist. Specifically, the link types in the whitelist are processed in the following manner. The link-types are first sorted in descending order by number of links. For the highest ranked link type, the link type weight starts at $\frac{1}{M}$. Each subsequent link-type weight is then incrementally increased by $\frac{1}{M}$, until the lowest ranked link-type weight is one. In other words, a higher ranked link type is given a smaller link-type weight and is most likely a communal relationship.

3.3 Spike Detection

This section contrasts SD with CD; and presents the need for SD, in order to improve resilience and adaptivity.

Before proceeding with a description of SD, it is necessary to reinforce that CD finds real social relationships to reduce the suspicion score, and is tamper resistant to synthetic social relationships. It is the whitelist-oriented approach on a fixed set of attributes. In contrast, SD finds spikes to increase the suspicion score, and is probe resistant for attributes. Probe resistance reduces the chances a fraudster will discover attributes used in the SD score calculation. It is the attribute-oriented approach on a variable-size set of attributes. A side note: SD cannot use a whitelist-oriented approach because it was not designed to create multiattribute links on a fixed-size set of attributes.

CD has a fundamental weakness in its attribute threshold. Specifically, CD must match at least three values for our data set. With less than three matched values, our whitelist does not contain real social relationships because some values, such as given name and unit number, are not unique identifiers. The fraudster can duplicate one or two important values which CD cannot detect.

SD complements CD. The redundant attributes are either too sparse where no patterns can be detected, or too dense where no denser values can be found. The redundant attributes are continually filtered, only selected attributes in the form of not-too-sparse and not-too-dense attributes are used for the SD suspicion score. In this way, the exposure of the detection system to probing of attributes is reduced because only one or two attributes are adaptively selected.

Suppose there was a bank's marketing campaign to give attractive benefits for it's new ladies' platinum credit card. This will cause a spike in the number of legitimate credit card applications by women, which can be erroneously interpreted by the system as a fraudster attack.

To account for the changing legal behavior caused by external events, SD strengthens CD by providing attribute weights which reflect the degree of importance in attributes. The attributes are adaptive for CD in the sense that its attribute weights are continually determined. This addresses external events such as the entry of new organizations and exit of existing ones, and marketing campaigns of organizations which do not contain any patterns and are likely to cause three natural changes in attribute weights. These changes are volume drift where the overall volume fluctuates, population drift where the volume of both fraud and legal classes fluctuates independent of each other, and concept drift which involves changing legal characteristics that can become similar to fraud characteristics. By tuning attribute weights, the detection system makes allowance for these external events.

TABLE 4
Overview of Spike Detection Algorithm

Inputs
v_i (current application)
W number of v_j (moving window)
t (current step)
$T_{similarity}$ (string similarity threshold)
θ (time difference filter)
α (exponential smoothing factor)

Outputs
$S(v_i)$ (suspicion score)
w_k (attribute weight)

SD algorithm

Step 1: Single-step scaled counts [match v_i against W number of v_j to determine if a single value exceeds $T_{similarity}$ and its time difference exceeds θ]

Step 2: Single-value spike detection [calculate current value's score based on weighted average (using α) of t Step 1's scaled matches]

Step 3: Multiple-values score [calculate $S(v_i)$ from Step 2's value scores and Step 4's w_k]

Step 4: SD attributes selection [determine w_k for SD at end of g_x]

Step 5: CD attribute weights change [determine w_k for CD at end of g_x]

In general, SD trades off effectiveness (degrades security because it has more false positives without filtering out communal relationships and some data errors) for efficiency (improves computation speed because it does not match against the whitelist, and can compute each attribute in parallel on multiple workstations). In contrast, CD trades off efficiency (degrades computation speed) for effectiveness (improves security by accounting for communal relationships and more data errors).

3.4 SD Algorithm Design

This section explains how the SD algorithm works in real time with the CD algorithm, and in terms of its six inputs, two outputs, and five steps.

From the data stream point-of-view, using a series of window steps, the SD algorithm matches the current application's value against a moving window of previous applications' values. It calculates the current value's score by integrating all steps to find spikes. Then, it calculates the current application's score using all values' scores and attribute weights. Also, at the end of the current Mini-discrete data stream, the SD algorithm selects the attributes for the SD suspicion score, and updates the attribute weights for CD.

Table 4 shows the data input and five parameters.

- v_i: unscored current application (previously introduced in Section 3.1).
- W: In SD, it is a time-based window (such as previous 10 days).

- t: current step, also the number of steps in W.
- $T_{similarity}$: string similarity threshold between two values (previously described in Section 3.1).
- θ: time difference filter at the link level. It is a simplified version of the exact duplicate filter.
- α: In SD, it gradually discounts the effect of previous steps of each value as the older steps become less relevant.

Table 4 also shows the two outputs.

- $S(v_i)$: SD suspicion score of current application.
- w_k: In SD, each attribute weight is automatically updated at the end of the current Minidiscrete data stream.

While Table 4 gives an overview of the SD algorithm's five steps, the details in each step are presented below.

Step 1: Single-step scaled count. The first step of the SD algorithm matches every current value against a moving window of previous values in steps

$$a_{i,j} = \begin{cases} 1, & \text{if } Jaro-Winkler(a_{i,k}, a_{j,k}) \geq T_{similarity} \\ & \quad \text{and } Time(a_{i,k}, a_{j,k}) \geq \theta, \\ 0, & \text{otherwise,} \end{cases} \tag{8}$$

where $a_{i,j}$ is the single-attribute match between the current value and a previous value. The first case uses Jaro-Winkler(.) [30], which is case sensitive, and can also be cross-matched between current value and previous values from another similar attribute, and Time(.) which is the time difference in minutes. The second case is a nonmatch because the values are not similar, or recur too quickly

$$s_\tau(a_{i,k}) = \sum_{a_{j,k} \in L(a_{i,k})} \frac{a_{i,j}}{\kappa}, \tag{9}$$

where $s_\tau(a_{i,k})$ represents the scaled matches in each step (the moving window is made up of many steps) to remove volume effects. $L(a_{i,k})$ is the set of previous values within each step which the current value matches, and κ is the number of values in each step.

Step 2: Single-value spike detection. The second step of the SD algorithm is the calculation of every current value's score by integrating all steps to find spikes. The previous steps act as the established baseline level

$$S(a_{i,k}) = (1-\alpha) \times s_t(a_{i,k}) + \alpha \times \frac{\sum_{\tau=1}^{t-1} s_\tau(a_{i,k})}{t-1}, \tag{10}$$

where $S(a_{i,k})$ is the current value score.

Step 3: Multiple-values score. The third step of the SD algorithm is the calculation of every current application's score using all values' scores and attribute weights

$$S(v_i) = \sum_{k=1}^{N} S(a_{i,k}) \times w_k, \tag{11}$$

where $S(v_i)$ is the SD suspicion score of the current application.

Step 4: SD attributes selection. At the end of every current Minidiscrete data stream, the fourth step of the SD algorithm selects the attributes for the SD suspicion score. This also highlights the probe-reduction of selected attributes

$$w_k = \begin{cases} 1, & \text{if } \dfrac{1}{2 \times N} \leq \dfrac{\sum_{i=1}^{p \times q} S(a_{i,k})}{i \times \sum_{k=1}^{N} w_k} \\ & \quad \leq \frac{1}{N} + \sqrt{\frac{1}{N} \times \sum_{k=1}^{N} (w_k - \frac{1}{N})^2}, \\ 0, & \text{otherwise,} \end{cases} \tag{12}$$

where w_k is the SD attribute weight applied to the SD attributes in (11). The first case is the average density of each attribute, or the sum of all value scores within a Minidiscrete stream for one attribute, relative to all other applications and attribute weights. In addition, the first case retains only the best attributes' weights within the lower-bound (half of default weight) and upperbound (default weight plus one standard deviation), by setting redundant attributes' weights to zero.

Step 5: CD attribute weights change. At the end of every current Minidiscrete data stream, the fifth step of the SD algorithm updates the attribute weights for CD

$$w_k = \frac{\sum_{i=1}^{p \times q} S(a_{i,k})}{i \times \sum_{k=1}^{N} w_k}, \tag{13}$$

where w_k is the SD attribute weight applied to the CD attributes in (3).

Standalone CD assumes all attributes are of equal importance. The resilient combination of CD-SD means that CD is provided attribute weights by SD, and these attribute weights reflect degree of importance in attributes. This is how CD and SD scores are combined to give a single score.

4 Experimental Results

4.1 Identity Data—Real Application Data Set

Substantial identity crime can be found in private and commercial databases containing information collected about customers, employees, suppliers, and rule violators. The same situation occurs in public and government-regulated databases such as birth, death, patient and disease registries; taxpayers, residents' address, bankruptcy, and criminals lists.

To reduce identity crime, the most important textual identity attributes such as personal name, Social Security Number (SSN), Date-of-Birth (DoB), and address must be used. The following publications support this argument: Jonas [16] ranks SSN as most important, followed by personal name, DoB, and address. Jost [17] assigns highest weights to permanent attributes (such as SSN and DoB), followed by stable attributes (such as last name and state), and transient (or ever changing) attributes (such as mobile phone number and email address). Sweeney [27] states that DoB, gender, and postcode can uniquely identify more than 80 percent of the United States (US) population. Head [12] and Kursun et al. [20] regard name, gender, DoB, and address as the most important attributes. The most important identity attributes differ from database to database. They are least likely to be manipulated, and are easiest to collect and investigate. They also have the least missing values, least spelling, and transcription errors, and have no encrypted values.

Extra precaution had to be taken in this project since this is the first time, to the best of the researchers' knowledge,

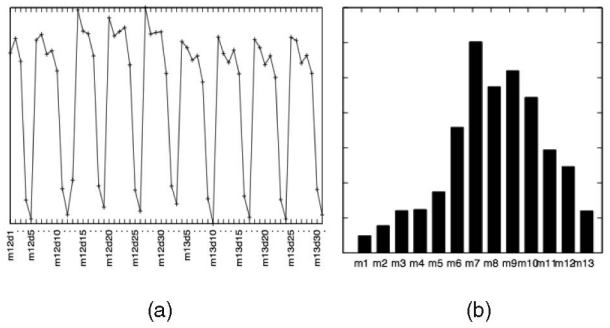


Fig. 1. Real Application Data Set (RADS). (a) Daily application volume for 2 months. (b) Fraud percentage across months.

that so much real identity data have been released for original credit application fraud detection research. Issues of privacy, confidentiality, and ethics were of prime concern.

This real data set was chosen because, at experimentation time, it had the most recent fraud behavior. Although this real data set cannot be made available, there is a synthetic data set of 50,000 credit applications which is available at https://sites.google.com/site/cliftonphua/communal-fraud-scoring-data.zip.

The specific summaries and basic statistics of the real credit application data are discussed below. For purposes of confidentiality, the application volume and fraud percentage in Fig. 1 have been deliberately removed. Also, the average fraud percentage (known fraud percentage in all applications) and specific attributes for application fraud detection cannot be revealed.

There are 13 months ($m1$ to $m13$) with several million applications. Each day ($d1$ to $d31$) has more than 10,000 applications. These historical data are unsampled, time-stamped to the milliseconds, and modeled as data streams. Fig. 1a illustrates that the detection system has to handle a more rapid and continuous data stream on weekdays than weekends.

There are about 30 raw attributes such as personal names, addresses, telephone numbers, driver license numbers (or SSN), DoB, and other identity attributes (but no link attribute). Only 19 of the most important identity attributes (I to XIX) are selected. All numerical attributes are treated as string attributes. Some of these identifying attributes, including names, were encrypted to preserve privacy. For our identity crime detection data, its encrypted attributes are limited to exact matching because the particular encryption method was not made known to us. But in a real application, homomorphic encryption [18] or unencrypted attributes would be used to allow string similarity matching. Another two problems are many missing values in some attributes, and hash collisions in encrypted attributes (different original values encrypted into the same encrypted value), but it is beyond the scope of this paper to present any solution.

The imbalanced class is extreme, with less than 1 percent of known frauds in all binary class-labeled (as "fraud" or "legal") applications. Fig. 1b depicts that known frauds are significantly understated in the provided applications. The main reason for fewer known frauds is having only 8 months ($m7$ to $m14$) of known frauds linked to 13 months of applications. Six months ($m1$ to $m6$) of known frauds were not provided. This results in $m6$ to $m10$ having the highest fraud percentage, but this is not true. Other reasons include some frauds which were unlabeled, having been inadvertently overlooked. Some known frauds are labeled once but not their duplicates, while some organizations do not contribute known frauds.

TABLE 5
Confusion Matrix

	Known frauds	Unknowns
Alerts	tp	fp
Non-alerts	fn	tn

The impact of fewer known frauds means algorithms will produce poorer results and lead to incorrect evaluation. To reduce this negative impact and improve scalability, the data have been rebalanced by retaining all known frauds but randomly undersampling unknown applications by 90 percent.

There are multiple sources, consisting of 31 organizations ($s1$ to $s31$) that provided the applications. Top-5 of these organizations ($s1$ to $s5$) can be considered large (with at least 10,000 applications per month), and more important than others, because they contribute more income to the credit bureau. Each organization contributes their own number and type of attributes.

The data quality was enhanced through the cleaning of two obvious data errors. First, a few organizations' applications, with slightly more than 10 percent of all applications, were filtered. This was because some important unstructured attributes were encrypted into just one value. Also, several "dummy" organizations' applications, comprising less than 2 percent of all applications, were filtered. They were actually test values particularly common in some months.

After the above data preprocessing activities, the actual experimental data provided significantly improved results. This was observed using the parameter settings in CD and SD (Section 4.3). These results have been omitted to focus on the results from CD and SD parameter settings and attributes.

In addition, which are the training and test data sets? The CD, SD, and classification algorithms use eight consecutive months ($m6$ to $m13$) out of 13 months data (each month is also known as a Minidiscrete stream in this paper) where known frauds are not significantly understated. For creating whitelist, selecting attributes, or setting attribute weights in the next month, the training set is the previous month's data. For evaluation, the test set is the current month's data. Both training and test data sets are separate from each other. For example, in CD, the initial whitelist is constructed from $m5$ training data, applied to $m6$ test data; and so on, until the final whitelist is constructed from $m12$ training data, and applied to $m13$ test data.

4.2 Evaluation Measure

Table 5 shows four main result categories for binary-class data with a given decision threshold. Alerts (or alarms) refer to applications with scores which exceed the decision

TABLE 6
Evaluation Measures

Measure	Description
precision	$\frac{tp}{tp+fp}$
recall, sensitivity	$\frac{tp}{tp+fn}$
(1 - specificity)	$\frac{fp}{fp+tn}$
F-measure curve	$\frac{2\times precision\times recall}{precision+recall}$ plotted against X thresholds
Receiver Operating Characteristic (ROC) curve	All scores are ranked in descending order; and sensitivity versus (1 - specificity) plotted against X thresholds

threshold, and subjected to responses such as human investigation or outright rejection. Nonalerts are applications with scores lower than the decision threshold. tp, fp, fn, and tn are the number of true positives (or hits), false positives (or false alarms), false negatives (or misses), and true negatives (or normals), respectively.

Table 6 briefly summarizes useful evaluation measures for scores. This paper uses the F-measure curve [31] and Receiver Operating Characteristic (ROC) curve [8] with 11 threshold values from zero to one to compare all experiments. The additional thresholds are needed because F-measure curves seldom dominate one another over all thresholds. The F-measure curve is recommended over other useful measures for the following reasons. First, for confidentiality reasons, precision-recall curves are not used as they will reveal true positives, false positives, and false negatives. Second, in imbalanced class data, ROC curves, AUC, and accuracy understates false positive percentage because they use true negatives [5]. Third, being a single-value measure, NTOP-k [4] does not evaluate results for more than one threshold. The ROC curve can be used to compliment F-measure curve because the former allows the reader to directly interpret if CD and SD are really reducing false positives. Reducing false positives is very important because staff costs for manual investigation, and valuable customers lost due to credit application rejection are very expensive.

In this paper, the scores have two unique characteristics. First, the CD score distribution is heavily skewed to the left, while SD score distribution is more skewed to the right. Most scores are zero as values are usually sparse. All the zero scores have been removed since they are not relevant to decision making. This will result in more realistic F-measures, although the number of applications in each F-measure will most likely be different. Second, some scores can exceed one since each application can be similar to many others. In contrast, classifier scores from naive Bayes, decision trees, logistic regression, or SVM exhibit a normal distribution and each score is between zero and one.

4.3 CD and SD's Experimental Design

All experiments were performed on a dedicated 2 Xeon Quad Core (8 2.0GHz CPUs) and 12 Gb RAM server, running on Windows Server 2008 platform. Communal and spike detection algorithms, as well as evaluation measures, were coded in Java. The application data were stored in a MySQL database. The plan here is to process all real applications from RADS with the most influential parameters and their values. These influential parameters are known to provide the best results based on the experience from hundreds of previous experiments. However, the best results are also dependent on setting the right value for each influential parameter in practice, as some parameters are sensitive to a change in their value.

There are seven experiments which focus on specific claims in this paper:

1. No-whitelist.
2. CD-baseline.
3. CD-adaptive.
4. SD-baseline.
5. SD-adaptive.
6. CD-SD-resilient.
7. CD-SD-resilient-best.

The first three experiments address how much the CD algorithm reduces false positives. The no-whitelist experiment uses zero link types ($M = 0$) to avoid using the whitelist. The CD-baseline experiment has the following parameter values (based on hundreds of previous CD experiments):

- $W =$ set to what is convenient for experimentation (for reasons of confidentiality, the actual W cannot be given)
- $M = 100$,
- $T_{similarity} = 0.8$,
- $T_{attribute} = 3$,
- $\eta = 120$, and
- $\alpha = 0.8$.

In other words, the CD-baseline uses a whitelist with 100 most frequent link types, and sets the string similarity threshold, attribute threshold, exact duplicate filter, and the exponential smoothing factor for scores. To validate the usefulness of the adaptive CD algorithm's changing parameter values, CD-adaptive experiment has three parameters (W, M, $T_{similarity}$) where their values can be changed according to the SoA.

The fourth and fifth experiments show if the SD algorithm increases power. The next experiment, SD-baseline, has the following parameter values (based on hundreds of previous SD experiments):

- $N = 19$,
- $t = 10$,
- $T_{similarity} = 0.8$,
- $\theta = 60$, and
- $\alpha = 0.8$.

In other words, the SD-baseline uses all 19 attributes, a moving window made up of 10 window steps, and sets string similarity threshold, time difference filter, and the exponential smoothing factor for steps. The SD-adaptive experiment selects two best attributes for its suspicion score.

The last two experiments highlight how well the CD-SD combination works. The CD-SD-resilient experiment is actually CD-baseline which uses attribute weights provided by SD-baseline. To empirically evaluate the detection system, the final experiment is CD-SD-resilient-best

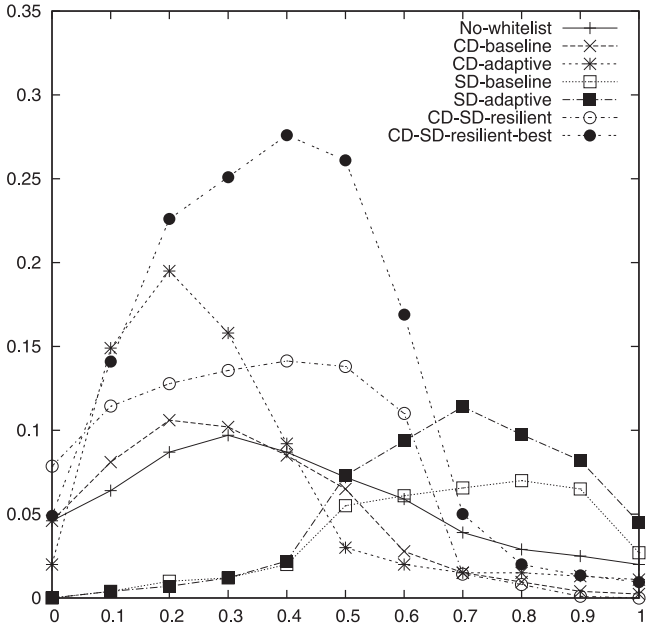


Fig. 2. F-measure curves of CD and SD experiments.

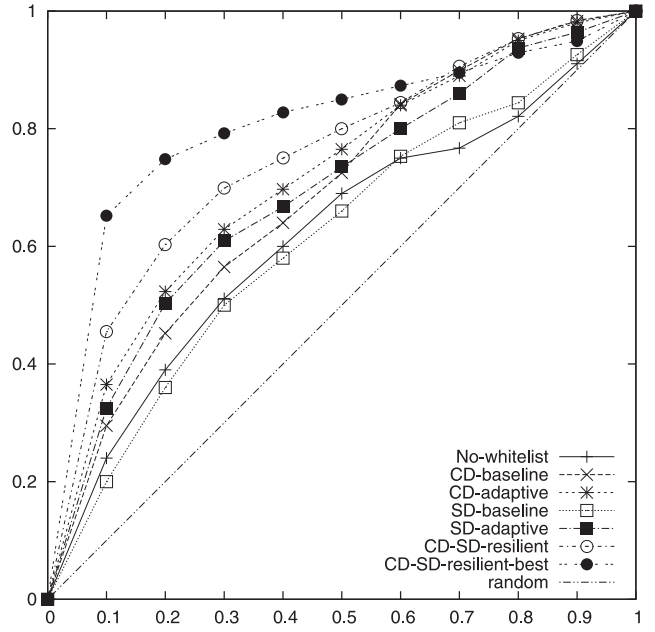


Fig. 3. ROC curves of CD and SD experiments.

experiment with the best parameter setting below (without adaptive CD algorithm's changing parameter values):

- $W =$ set to what is expected to be used in practice,
- $T_{similarity} = 1$,
- $T_{attribute} = 4$, and
- SD attribute weights.

4.4 CD and SD's Results and Discussion

The CD F-measure curves skew to the left. The CD-related F-measure curves start from 0.04 to 0.06 at threshold 0, and peak from 0.08 to 0.25 at thresholds 0.2 or 0.3. On the other hand, the SD F-measure curves skew to the right.

Without the whitelist, the results are inferior. From Fig. 2 at threshold 0.2, the no-whitelist experiment (F-measure below 0.09) performs poorer than the CD-baseline experiment (F-measure above 0.1). From Fig. 3, the no-whitelist experiment has about 10 percent more false positives than the CD-baseline experiment. This verifies the hypothesis that the whitelist is crucial because it reduces the scores of these legal behavior and false positives; also, the larger the volume for a link type, the higher the probability of a communal relationship.

From Fig. 2 at threshold 0.2, the CD-adaptive experiment (F-measure above 0.16) has a significantly higher F-measure than the CD-baseline experiment. From Fig. 3, the CD-adaptive experiment has about 5 percent less false positives in the early part of the ROC curve than the CD-baseline experiment. The interpretation is that the most useful parameters are moving window and number of link types. More importantly, the adaptive CD algorithm finds the balance between effectiveness and efficiency to produce significantly better results than the CD-baseline experiment. This empirical evidence suggests that there is tamper resistance in parameters and the whitelist as some parameters' values and whitelist's link types are changed in a principled fashion.

From Fig. 2 at threshold 0.7, the SD-adaptive experiment (F-measure around 0.1) has a significantly higher F-measure than the SD-baseline experiment. Also, SD-adaptive experiment has almost the same F-measure as the CD-baseline experiment but at different thresholds. Since most attributes are redundant, the adaptive SD algorithm only needs to select the two best attributes for calculation of the suspicion score. This means that the adaptive SD algorithm on two best attributes produces better results than the SD algorithm on all attributes, as well as similar results to the basic CD algorithm on all attributes.

Across thresholds 0.2 to 0.5, the CD-SD-resilient-best experiment (F-measure above 0.23) has a F-measure which is more than twice the CD-baseline experiment's. This is the most apparent outcome of all experiments: the CD algorithm, strengthened by the SD algorithm's attribute weights, and with the right parameter setting, delivers superior results, despite an extremely imbalanced class (at least for the given data set). In addition, results from the CD-SD-resilient experiment supports the view that SD attribute weights strengthen the CD algorithm; and resilience (CD-SD-resilience) is shown to be better than adaptivity (CD-adaptive and SD-adaptive).

Extending CD-SD-resilient-best experiment, Fig. 4 shows the results of doubling the most influential parameters' values. W and $T_{attribute}$ have significant increases in F-measure over most thresholds, and M has a slight increase at thresholds 0.2 to 0.4.

Results on the data support the argument that successful credit application fraud patterns are characterized by sudden and sharp spikes in duplicates. However, this result is based on some assumptions and conditions shown by the research to be critical for effective detection. A larger moving window and attribute threshold, as well as exact matching and the whitelist must be used. There must also be tamper resistance in parameters and the whitelist. It is also assumed that SD attribute weights are used for SD attributes

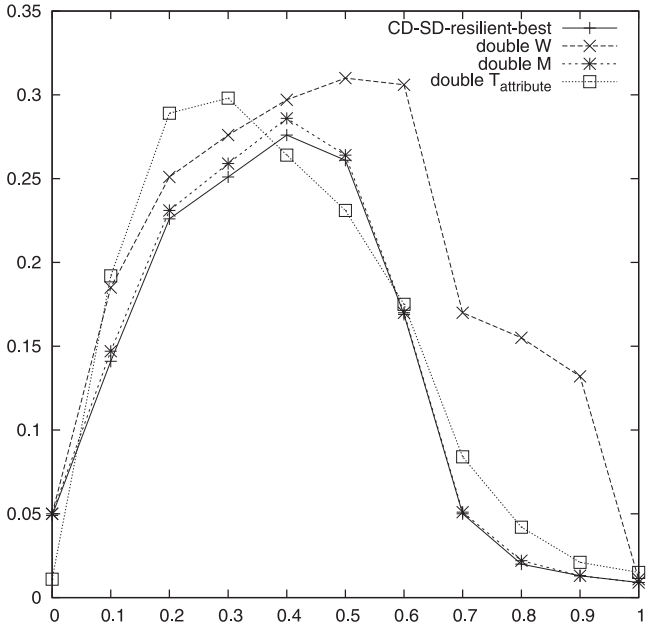


Fig. 4. F-measure curves of CD-SD-resilient-best parameters.

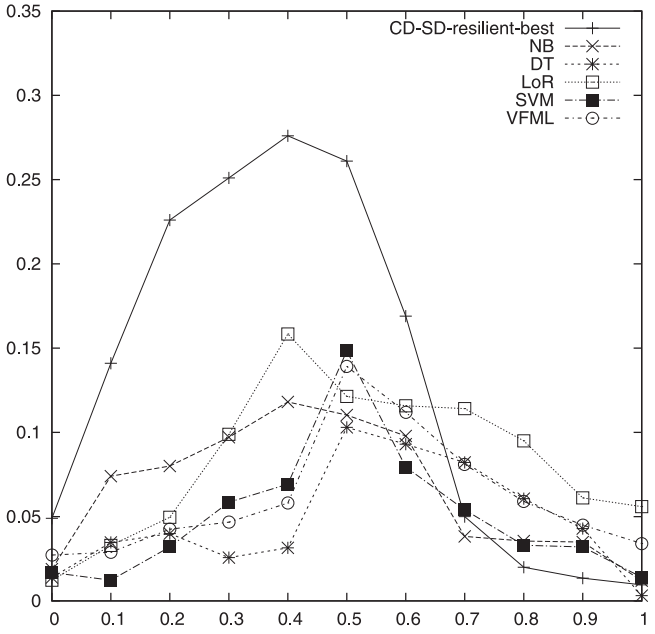


Fig. 6. F-measure curves of five classification algorithms.

selection (probereduction of attributes), and SD attribute weights are used for CD attribute weights change. However, the results can be slightly incorrect because of the encryption of some attributes and the significantly understated number of known frauds. Also, the solutions could not account for the effects of the existing defences—business rules and scorecards, and known fraud matching—on the results.

4.5 Drilled-Down Results and Discussion

The CD-SD-resilient-best experiment shows that the CD-SD combination method works best for all 31 organizations as a whole. The same method may not work well for every organization. Fig. 5 shows the detailed breakdown of top-5 organizations' ($s1$ to $s5$) results from the CD-SD-resilient-best experiment. Similar to the CD-SD-resilient-best experiment, the top-5 organizations' F-measure curves are skewed to the left.

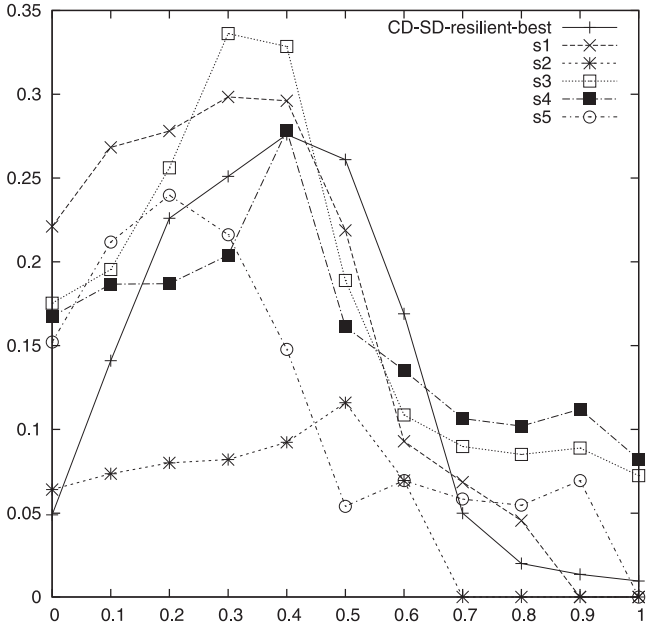


Fig. 5. F-measure curves of top-5 organizations.

Across thresholds 0.2 to 0.5, two organizations, $s1$ (F-measure above 0.22) and $s3$ (F-measure above 0.19) have comparable F-measures than the CD-SD-resilient-best experiment (F-measure above 0.23). In contrast, for the same thresholds, three organizations, $s4$ (F-measure above 0.16), $s2$ (F-measure above 0.08), and $s5$ (F-measure above 0.05) have significantly lower F-measures. However, in CD-baseline experiment, for thresholds 0.2 to 0.5, $s5$ performs better than $s4$. This implies that most methods or parameter settings can work well for only some organizations.

4.6 Classifier-Comparison Experimental Design, Results, and Discussion

Are classification algorithms suitable for the real-time credit application fraud detection domain? To answer the above question, four popular classification algorithms with default parameters in WEKA [29] were chosen for classifier experiments. The algorithms were Naive Bayes (NB), C4.5 Decision Tree (DT), Logistic Regression (LoR), and Support Vector Machines—current state-of-the-art libSVM. A well-known data stream classification algorithm, Very Fast Machine Learner (VFML) which is a Hoeffding decision tree, is also used with default parameters in MOA [1]. They were applied to the same training and test data used by CD and SD algorithms, and there was an extra step to convert the string attributes to word vector ones. The following experiments assume that ground truth is available at training time (see Section 1.2 for a description of the problems in using known frauds).

Classification algorithms are not the most accurate and scalable for this domain. Fig. 6 compares the five classifiers against CD-SD-resilient-best experiment with F-measure across 11 thresholds. Across thresholds 0.2 to 0.5, CD-SD-resilient-best experiment's F-measure can be several times higher than the five classifiers: NB (F-measure above 0.08), LoR (F-measure above 0.05), VFML (F-measure above 0.04),

TABLE 7
Relative Time of Five Classification Algorithms

Experiment(s)	Relative time
CD-SD-resilient-best	1
NB	1.25
DT	5
VFML	18
LoR	60
SVM	156

SVM, and DT (F-measure above 0.03). Also, results did not improve from training the five classifiers on labeled multi-attribute links, and applying the classifiers to multiattribute links in the test data. Table 7 measures relative time of five classifiers using CD-SD-resilient-best experiment as baseline. Time refers to total system time for the algorithm to complete. CD-SD-resilient-best experiment is orders of magnitude faster than the classifier experiments because it does not need to train on many word vector attributes and with few known frauds.

5 Conclusion

The main focus of this paper is **Resilient Identity Crime Detection**; in other words, the real-time search for patterns in a multilayered and principled fashion, to safeguard credit applications at the first stage of the credit life cycle.

This paper describes an important domain that has many problems relevant to other data mining research. It has documented the development and evaluation in the data mining layers of defence for a real-time credit application fraud detection system. In doing so, this research produced three concepts (or "force multipliers") which dramatically increase the detection system's effectiveness (at the expense of some efficiency). These concepts are resilience (multi-layer defence), adaptivity (accounts for changing fraud and legal behavior), and quality data (real-time removal of data errors). These concepts are fundamental to the design, implementation, and evaluation of all fraud detection, adversarial-related detection, and identity crime-related detection systems.

The implementation of CD and SD algorithms is practical because these algorithms are designed for actual use to complement the existing detection system. Nevertheless, there are limitations. The first limitation is effectiveness, as scalability issues, extreme imbalanced class, and time constraints dictated the use of rebalanced data in this paper. The counter-argument is that, in practice, the algorithms can search with a significantly larger moving window, number of link types in the whitelist, and number of attributes. The second limitation is in demonstrating the notion of adaptivity. While in the experiments, CD and SD are updated after every period, it is not a true evaluation as the fraudsters do not get a chance to react and change their strategy in response to CD and SD as would occur if they were deployed in real life (experiments were performed on historical data).

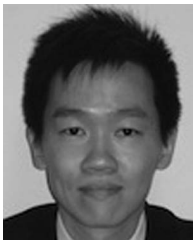
Acknowledgments

Thanks to Dr. Warwick Graco and Mr. Kelvin Sim for insightful comments; Australian Research Council (ARC) for funding under Linkage Grant Number LP0454077.

References

[1] A. Bifet and R. Kirkby Massive Online Analysis, Technical Manual, Univ. of Waikato, 2009.
[2] R. Bolton and D. Hand, "Unsupervised Profiling Methods for Fraud Detection," *Statistical Science,* vol. 17, no. 3, pp. 235-255, 2001.
[3] P. Brockett, R. Derrig, L. Golden, A. Levine, and M. Alpert, "Fraud Classification Using Principal Component Analysis of RIDITs," *The J. Risk and Insurance,* vol. 69, no. 3, pp. 341-371, 2002, doi: 10.1111/1539-6975.00027.
[4] R. Caruana and A. Niculescu-Mizil, "Data Mining in Metric Space: An Empirical Analysis of Supervised Learning Performance Criteria," *Proc. 10th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '04),* 2004, doi: 10.1145/1014052.1014063.
[5] P. Christen and K. Goiser, "Quality and Complexity Measures for Data Linkage and Deduplication," *Quality Measures in Data Mining,* F. Guillet and H. Hamilton, eds., vol. 43, Springer, 2007, doi: 10.1007/978-3-540-44918-8.
[6] C. Cortes, D. Pregibon, and C. Volinsky, "Computational Methods for Dynamic Graphs," *J. Computational and Graphical Statistics,* vol. 12, no. 4, pp. 950-970, 2003, doi: 10.1198/1061860032742.
[7] Experian. Experian Detect: Application Fraud Prevention System, Whitepaper, http://www.experian.com/products/pdf/experian_detect.pdf, 2008.
[8] T. Fawcett, "An Introduction to ROC Analysis," *Pattern Recognition Letters,* vol. 27, pp. 861-874, 2006, doi: 10.1016/j.patrec. 2005.10.010.
[9] A. Goldenberg, G. Shmueli, R. Caruana, and S. Fienberg, "Early Statistical Detection of Anthrax Outbreaks by Tracking Over-the-Counter Medication Sales," *Proc. Nat'l Academy of Sciences USA (PNAS '02),* vol. 99, no. 8, pp. 5237-5240, 2002.
[10] G. Gordon, D. Rebovich, K. Choo, and J. Gordon, "Identity Fraud Trends and Patterns: Building a Data-Based Foundation for Proactive Enforcement," *Center for Identity Management and Information Protection,* Utica College, 2007.
[11] D. Hand, "Classifier Technology and the Illusion of Progress," *Statistical Science,* vol. 21, no. 1, pp. 1-15, 2006, doi: 10.1214/088342306000000060.
[12] B. Head, "Biometrics Gets in the Picture," Information Age, pp. 10-11, Aug.-Sept. 2006.
[13] L. Hutwagner, W. Thompson, G. Seeman, and T. Treadwell, "The Bioterrorism Preparedness and Response Early Aberration Reporting System (EARS)," *J. Urban Health,* vol. 80, pp. 89-96, 2006.
[14] IDAnalytics, "ID Score-Risk: Gain Greater Visibility into Individual Identity Risk," Unpublished, 2008.
[15] M. Jackson, A. Baer, I. Painter, and J. Duchin, "A Simulation Study Comparing Aberration Detection Algorithms for Syndromic Surveillance," *BMC Medical Informatics and Decision Making,* vol. 7, no. 6, 2007, doi: 10.1186/1472-6947-7-6.
[16] J. Jonas, "Non-Obvious Relationship Awareness (NORA)," *Proc. Identity Mashup,* 2006.
[17] M. Kantarcioglu, W. Jiang, and B. Malin, "A Privacy-Preserving Framework for Integrating Person-Specific Databases," *Proc. UNESCO Chair in Data Privacy Int'l Conf. Privacy in Statistical Databases (PSD '08),* pp. 298-314, 2008, doi: 10.1007/978-3-540-87471-3_25.
[18] J. Kleinberg, "Temporal Dynamics of On-Line Information Streams," *Data Stream Management: Processing High-Speed Data Streams,* M. Garofalakis, J. Gehrke, and R. Rastogi, eds., Springer, 2005.
[19] O. Kursun, A. Koufakou, B. Chen, M. Georgiopoulos, K. Reynolds, and R. Eaglin, "A Dictionary-Based Approach to Fast and Accurate Name Matching in Large Law Enforcement Databases," *Proc. IEEE Int'l Conf. Intelligence and Security Informatics (ISI '06),* pp. 72-82, 2006, doi: 10.1007/11760146.
[20] J. Neville, O. Simsek, D. Jensen, J. Komoroske, K. Palmer, and H. Goldberg, "Using Relational Knowledge Discovery to Prevent Securities Fraud," *Proc. 11th ACM SIGKDD Int'l Conf. Knowledge Discovery in Data Mining (KDD '05),* 2005, doi: 10.1145/1081870.1081922.
[21] T. Oscherwitz, "Synthetic Identity Fraud: Unseen Identity Challenge," *Bank Security News,* vol. 3, p. 7, 2005.
[22] S. Roberts, "Control-Charts-Tests Based on Geometric Moving Averages," *Technometrics,* vol. 1, pp. 239-250, 1959.
[23] S. Romanosky, R. Sharp, and A. Acquisti, "Data Breaches and Identity Theft: When Is Mandatory Disclosure Optimal?," *Proc. Ninth Workshop Economics of Information Security (WEIS),* 2010.

[24] B. Schneier, *Beyond Fear: Thinking Sensibly about Security in an Uncertain World.* Copernicus, 2003.
[25] B. Schneier,, *Schneier on Security.* Wiley, 2008.
[26] L. Sweeney, "*k*-Anonymity: A Model for Protecting Privacy," *Int'l J. Uncertainty,* vol. 10, no. 5, pp. 557-570, 2002.
[27] R. Wheeler and S. Aitken, "Multiple Algorithms for Fraud Detection," *Knowledge-Based Systems,* vol. 13, no. 3, pp. 93-99, 2000, doi: 10.1016/S0950-7051(00)00050-2.
[28] W. Winkler, "Overview of Record Linkage and Current Research Directions," Technical Report RR 2006-2, US Census Bureau, 2006.
[29] I. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java.* Morgan Kauffman, 2000.
[30] W. Wong, "Data Mining for Early Disease Outbreak Detection," PhD thesis, Carnegie Mellon Univ., 2004.
[31] W. Wong, A. Moore, G. Cooper, and M. Wagner, "Bayesian Network Anomaly Pattern Detection for Detecting Disease Outbreaks," *Proc. 20th Int'l Conf. Machine Learning (ICML '03),* pp. 808-815, 2003.



Clifton Phua is a scientist working on security and healthcare-related data mining at the Institute of Infocomm Research (I^2R), Singapore. He is a member of the IEEE.

Kate Smith-Miles is a professor and head of the School of Mathematical Sciences at Monash University, Australia. Her current research interests include neural networks, intelligent systems, and data mining. She is a senior member of the IEEE.

Vincent Cheng-Siong Lee is an associate professor in the Clayton School of Information Technology, Monash University, Australia. His current research interests include data and text mining for business intelligence.

Ross Gayler is a senior research and development consultant working on credit scoring at Veda Advantage, Australia.