FraudBuster: Temporal Analysis and Detection of Advanced Financial Frauds

Michele Carminati[1], Alessandro Baggio[1], Federico Maggi[1,2]
Umberto Spagnolini[1], and Stefano Zanero[1]

[1] DEIB, Politecnico di Milano, Italy
[2] Trend Micro Inc.
{michele.carminati,federico.maggi,umberto.spagnolini,stefano.zanero}@polimi.it
alessandro1.baggio@mail.polimi.it

Abstract Modern financial frauds are frequently automated through specialized malware that hijacks money transfers from the victim's computer. An insidious type of fraud consists in repeatedly stealing small amounts of funds over time. A reliable detection of these fraud schemes requires an accurate modeling of the user's spending pattern over time. In this paper, we propose *FraudBuster*, a framework that exploits the end user's recurrent vs. non-recurrent spending pattern to detect these sophisticated frauds. *FraudBuster* is based on a learning stage that builds, for each user, temporal profiles and quantifies the deviation of each incoming transaction from the learned model. The final output is the aggregated score that quantifies the risk of a user of being defrauded. In this setting, *FraudBuster* detects frauds as transactions that are not simply "anomalous", but that would change the user's spending profile.
We deployed *FraudBuster* in the real-world setting of a national banking group and measured the detection performance, showing that it can outperform existing solutions.

1 Introduction

Financial frauds have been steadily increasing over the past few years, resulting in billions of dollar losses [1]. Malware seems to be evolving through the collaboration between malware creators, growing by 16% since 2016. In 2016 financial malware infected about 2,8 million personal devices, a 40% increase since 2015 [2]. Despite financial institutions rely on fraud-analysis systems, fraudsters keep refining their techniques to remain unaccountable. Automated frauds are typically implemented via specialized malware, sold in underground markets, that can be easily customized to perform and/or hijack money transfers. An insidious type of fraud consists in keeping a "low profile" by stealing small amounts of funds in multiple rounds over time. Due to their stealthiness and recurring nature, we call these attacks as *salami-slicing* frauds, referring to the well-known fraudulent technique [3]. Moreover, Internet banking seems like the perfect venue for this type of attacks, due to the increasing adoption of micro-payment systems, with direct debit on the bank account. Detecting these sophisticated schemes requires a robust modeling of the end user's spending patterns to exclude false positives due to legitimate, small-amount, recurrent transfers (e.g., subscriptions). The detection task is challenging because frauds are dynamic and "blend in" with legitimate transactions. Furthermore, the scarcity of publicly available, real-world datasets makes research in this area a daunting task. Our *state-of-the-art* analysis revealed that existing works presume the existence of periodicities in users' spending patterns, without verifying it on real data.

In this paper, we propose *FraudBuster*, a fraud-analysis system that aims to detect *salami-slicing* frauds by exploiting a precise modeling of recurrent vs. non-recurrent spending patterns. *FraudBuster* is based on a learning stage that automatically estimates the end user's temporal profiles by means of historical (and likely fraud-free) spending patterns and quantifies the deviation of the current user's spending profile from the learned model. In particular, we apply signal processing techniques to extract temporal patterns "hidden" in the time series obtained from the transaction history. We show that temporal patterns exist, and thus a fraud-detection technique augmented by temporal-pattern classification is more effective than conventional detection approaches. First, *FraudBuster* classify each user based on its spending pattern. Then if a user is labeled as "periodic" (i.e., his or hers spending patter is characterized by periodicity), *FraudBuster* aligns and averages the observed time series to create a reference pattern model. In other words, we derive the most likely spending activity. Small deviations from strict periodic pattern are accounted by the dynamic time warping (DTW) algorithm [4] that is adapted here to measure the similarity between two temporal sequences up to a small deviation from the reference pattern. For both users with and without periodic spending patterns, *FraudBuster* uses a proper time-windowing analysis of transactions. For each incoming transaction, *FraudBuster* measures the deviation (i.e., anomaly score) of the user's spending activity from the learned model. The final output is the aggregated score that quantifies the risk of a user of being defrauded. By doing this, *FraudBuster* supports the analysts' ex-post analysis (i.e., manual investigation of frauds), making analysts focusing only on highly ranked users and on transactions that deviates most from the user's spending pattern.

We tested *FraudBuster* in the real-world context of a large national bank. Leveraging the domain expert's knowledge, we reproduced *salami-slicing* frauds performed against banking users. *FraudBuster* achieves a detection rate remarkably above *state-of-the-art* approaches, detecting 60% more defrauded users. In addition, we prove the effectiveness of the *time-series* analysis on the detection performance and investigate the robustness of *FraudBuster* against mimicry attacks and real-world *salami-slicing* frauds.

The main contributions are:

- We conduct a case-study analysis on a real-world dataset to show the importance of modeling recurrent vs. non-recurrent spending habits.
- We design *FraudBuster*, an efficient tool to detect *salami-slicing* frauds based on an accurate modeling of the user's temporal profile.
- We provide a comprehensive evaluation of *FraudBuster* and show that it outperforms the state of the art.

2 Background and Motivation

Internet banking services are heavily targeted by cyber criminals. A compromised banking account can be used to directly steal funds from the available balance or can be sold on the underground market. The main tools exploited by fraudsters are the so-called "banking Trojans" or "infostealers," specific types of malware that leverage Man-in-the-Browser (MitB) techniques to intercept and modify web pages, as well as transaction content, at the level of the rendering engine of the browser, in a fully transparent way. Endpoint solutions offer little protection because they are hard to deploy uniformly, due to the variety of devices. Also, modern infostealers are often able to bypass the second factor authentication, if present, by infecting the mobile device and stealing the OTPs. Therefore, effective fraud-detection solutions to identify fraudulent transfers are still a much-needed product. In the context under analysis, frauds can be considered as anomalies in banking transactions. State-of-the-art detection systems rely on statistical and data-mining techniques to detect anomalous transactions and support the analysts during manual investigation. We can broadly distinguish between three main cases of interest. First, a transfer can be anomalous *per se*. For instance, a transfer of two or three orders of magnitude above the maximum amount ever transferred by a given user is clearly anomalous. Secondly, a transaction can be anomalous within a certain *context* (e.g., day, week, time of the year). Third, a *series* of non-anomalous transactions could be anomalous only when observed all together. These are called *collective* anomalies and, according to [5], are the most challenging to detect. Whenever a modern banking Trojan infects the victim's machine, it can execute multiple transactions, either piggybacking while the user is already performing online banking tasks (i.e., transaction hijacking), or simply while the browser is open (e.g., via session stealing). This allows the fraudster to keep a "low-profile" by transferring small amounts of funds in multiple rounds over time, and thus evading even the most advanced detection techniques. Nowadays collective anomalies are growing with the aid of sophisticated malware kits [6]. We use the term *salami-slicing* frauds, referring to the well-known fraudulent attack scheme [3], to indicate collective frauds perpetrated over a given time span (e.g., day, week, month) with the goal of automatically stealing substantial amount of funds from the victims.

2.1 Goals and Challenges

The goal of this work is to propose a practical answer to the aforementioned problem, so as to support the banking analyst dealing with *salami-slicing* frauds. Due to the strict dependency between these frauds and time, the problem can be reduced to an anomaly detection on time series. However, it is difficult to distinguish between users with vs. without periodic spending patterns. In fact, time series obtained from money transfers are characterized by stochastic attribute values and shifts in time, which may conceal the true periodicity. Another challenging aspect is that frauds are dynamic, resembling legitimate transfers (e.g., when performed using the victim 's computers as a proxy), rare, and hidden in each user's transactions. Hence, it is hard to detect frauds with real-world-compliant precision and efficiency.

3 Related Works

Fraud and anomaly detection are a wide research topic for which we refer the reader to [5,7,8]. In this section, we focus on fraud detection from a temporal point of view, showing that the current landscape of fraud-detection solutions is not sufficiently mature to detect stealthy attacks that, similarly to *salami-slicing* frauds, are performed with the goal of evading detection techniques. In particular, we analyze the banking, intrusion detection, and credit card fraud detection contexts that shares many common aspects and, hence, can be compared.

Internet banking. One of the most recent and effective fraud-detection system for Internet banking is BankSealer [9]. It approaches the problem of detecting frauds from a temporal point of view by tailoring a set of thresholds on time-dependent attributes (e.g., number of transactions, the total amount, and maximum daily number of transactions per time windows). This technique models the spending habits in a simple yet effective way: a transaction is anomalous, in a certain time interval, if exceeds the set of detecting thresholds. However, BankSealer does not consider whether transactions exhibit a periodic and repeating pattern in a reference period (e.g., monthly, weekly, daily). This is reflected in a low detection rate when multiple low-amount frauds (like *salami-slicing* frauds) are observed. The unsupervised approach presented in [1] applies contrast pattern mining considering the dependence between events at different points in time to detect not fraudulent transactions but anomalous activities in the interaction between the bank's customer and the web application.

Intrusion Detection. In [10] and [11], the authors apply the Discrete Fourier Transform (DFT) on time series to identify anomalous frequencies. In particular, [11] compares the frequency representations by means of Mutual Information [12] to detect anomalies. In the same direction, [13] proposes a novel distance measure to compare two different DFT. However, these works rely on the assumption, not valid in our context, that the normal spectrum, derived from the network time series, is "flat", while the spectrum produced by an attack shows peak for specific frequencies.

Credit Card Fraud Detection. The authors of [14] propose time-based techniques applying a modified version of the k-means clustering algorithm to group transactions in each time window. The centroids of the resulting clusters (e.g., daily, weekly, and monthly cluster) are then used as baselines, and each new transaction's anomaly is quantified as the distance from such centroids. In [15], the authors compare well-known supervised techniques such as support vector machines, random forests and logistic regression applied on a real dataset. To consider time dependent features, they derive attributes by aggregating transactions, employing different time windows length. In [16], the authors aggregate transactions based on time windows of fixed length to extract time-dependent features. Then, they compute the anomaly score of a transaction with a logistic score. In [17], the authors employ the Basic Local Alignment Search Tool (BLAST) [18], a sequence alignment technique that identifies transactions that do not follow the spending pattern of the customers.

3.1 Critics

Existing works face the temporal anomaly detection problem by aggregating data instances in time windows and creating new time-dependent features, or by comparing past time series with the current one by means of the DFT. The main issue is that these methods assume that users exhibit periodic spending habits without demonstrating it. Another issue of the presented works is that they usually consider only few attributes for building user's profile. As demonstrated by our work, a deeper analysis of transaction's feature can be more effective. In addition, the authors in [1] show that classic time-based anomaly-detection methods (e.g., DFT) fail when applied *as is* in the online banking context, which is characterized by mixed scenarios, with users changing their habits over time. Beside this, the presented approaches based on DFT are interesting, but they usually assume an uncorrelated behavior for legitimate user (i.e., flat spectrum) and a "spiked" spectrum for the attacks. As shown in § 4, this assumption does not hold up in our context. A final critic is the high complexity of existing works. They often require a temporal complexity that scales with mn, where n is the length of the historical time series in which we want to search a subsequence with length m. For a real-time analysis, it is impracticable since n is very high. The clustering algorithm proposed in [14] is also infeasible since it considers all the possible permutations in the time windows to solve the shift of transactions.

4 Case Study

4.1 Recurrent Spending Patterns Analysis

Since we found no work that verified the actual presence of periodicities in user's spending patterns we run a case study. We performed a temporal analysis of real banking transactions, looking for recurrent spending patterns. This analysis is an essential prerequisite for detecting *salami-slicing* frauds effectively. Our goal is to distinguish between *periodic* and *non-periodic* spending patterns. We focus on a "per user" temporal analysis, grouping bank customers and representing

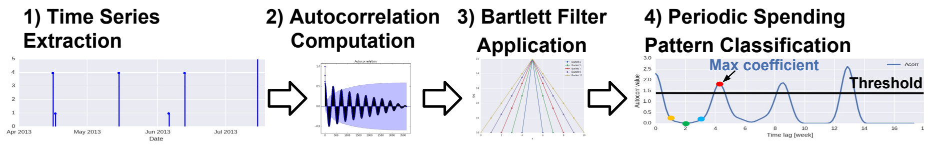

1) Time Series Extraction
2) Autocorrelation Computation
3) Bartlett Filter Application
4) Periodic Spending Pattern Classification
Apr 2013
May 2013
Jun 2013
Jul 2013
Date
Max coefficient
Threshold
Acorr
Time lag [week]

Figure 1. Time-series analysis and periodic spending pattern classification.

their activity with the time series of the daily number of transactions. These time series are characterized by isolated "spikes" (i.e., days in which the user performs transactions), interleaved by long idle periods. Moreover, we notice a minority of low peaks, which can be interpreted as noise, due to the intrinsic randomness of users' activity. As expected, time series present alignment problems between "corresponding" peaks (i.e., events belonging to different time series but "near" from the temporal point of view), due to the shifts in time of user's transactions. These make the analysis a challenging task.

Time Series Analysis and Periodic Spending Pattern Classification. We consider weekly, bi-weekly, three-weekly, and monthly periodic spending patterns. The selection of "n-weekly" users comes from the need of finding the smallest reasonable recurrent spending pattern (i.e., periodicity), excluding sub-periodicities; i.e., we do not want to consider as monthly users, users which are simply "weekly users" (a month is a multiple of a week and, hence, a weekly user is also monthly users, but not vice-versa). We do not consider longer periods due to the limited timespan of the dataset under analysis. To spot recurrent transactions we rely on the autocorrelation analysis [19,20]. The correlation of a time series with itself at different mutual time-shifts measures their similarity and a high autocorrelation value at given time lag τ implies a temporal pattern with periodicity $T = \tau$. To limit the problem of small shifts in user activity, we smooth the autocorrelation values by applying a convolution with a triangular-shaped Bartlett filter [21] that spreads each transaction spike over its nearby days, implicitly giving a time-tolerance in pattern analysis. With respect to other methods such as DFT or semblance analysis (see [20] for common techniques used in the time series domain), we choose to use Bartlett filter and Autocorrelation since, after an empirical evaluation, they best manage time series characterized by spikes and noise without generating too much ripple in the frequency domain. In order to label a user as "periodic" or "non-periodic", we calculate all the autocorrelation coefficients for $\tau =$\{weekly, bi-weekly, three-weekly, monthly\}. Then, we test each user periodic behavior based on the largest autocorrelation coefficient. To distinguish periodic from non-periodic users, we set a threshold: if no autocorrelation coefficient overcomes this value, the user is labeled as non-periodic. The threshold is defined from a percentage of the autocorrelation value in the origin (i.e., for $\tau = 0$), as this sets the reference value for self-similarity analysis. We empirically choose the threshold as the value that minimizes the miss-classification probability of periodic users, expressed in terms of the Minimum Distance of Pair Assignment (MDPA) [22]. The lower the MDPA value is, the closer relation between users exists and hence a common periodicity in their spending activity. This happens for a threshold's value equal to 70%, which produces an abrupt change in the trend of the metric analysis. Fig. 1 shows the applied methodology.

Results. We apply the aforementioned algorithm on the dataset under analysis. For this test, we keep users with at least one transaction per month in order to reduce the artifacts caused by occasional users. As shown in Tab. 1, most of users (about 56%) do not exhibit any measurable temporal pattern. However, a

Table 1. Periodic spending pattern classification results.

Dataset	**Classified Periodic Spending Pattern (%)**				
	Weekly	Bi-Weekly	Three-Weekly	Monthly	No Periodicity
April - July	2.7	4.0	5.2	**32.0**	56.1
April - August	2.5	3.2	4.0	**32.8**	57.5

non-negligible portion of users, about 32% exhibits a monthly periodic spending pattern. If we consider also "n-weekly" users, the total number of periodic spending patterns reaches the 44%. Hence, the percentage of weekly, bi-weekly and three-weekly groups is quite small. From a manual inspection, we noticed that these groups are more subject to noise. This is also proved by the fact that their number largely decrease when considering longer timespan. Since we cannot prove with empirical evidence that these groups are statistically meaningful, we consider only the monthly periodicity. Finally, we test the classification algorithm on synthetic users, with different time span length. This analysis allows us to verify the quality of our classification algorithm, since it provides a ground truth and shows how the classification changes varying parameters and the timespan length. We run several tests, each one on 1,000 synthetic users, varying the time series length and the probability of a user to be *active* on monthly basis. For each of these configuration, we analyze the percentage of monthly users classified. With a longer time span, the percentage of non-periodic users identified as monthly users decreases, while the percentage of monthly users, identified as periodic, increases. This because in short time series the presence of noise and time shifts are much more relevant and leads to a wrong users' classification.

5 Approach Overview

In this section, we describe *FraudBuster*, an analysis system that detects *salami-slicing* frauds by exploiting recurrent vs. non-recurrent spending patterns.

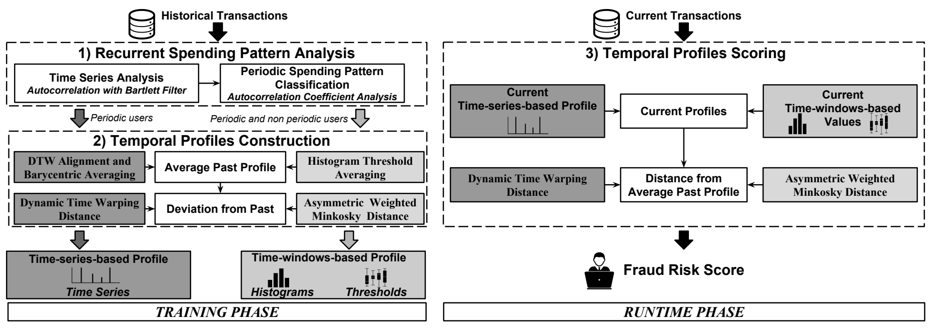

Historical Transactions
1) Recurrent Spending Pattern Analysis
Time Series Analysis
Autocorrelation with Bartlett Filter
Periodic Spending Pattern Classification
Autocorrelation Coefficient Analysis
Periodic users
Periodic and non periodic users
2) Temporal Profiles Construction
DTW Alignment and Barycentric Averaging
Average Past Profile
Histogram Threshold Averaging
Dynamic Time Warping Distance
Deviation from Past
Asymmetric Weighted Minkosky Distance
Time-series-based Profile
Time Series
Time-windows-based Profile
Histograms
Thresholds
TRAINING PHASE
Current Transactions
3) Temporal Profiles Scoring
Current Time-series-based Profile
Current Profiles
Current Time-windows-based Values
Dynamic Time Warping Distance
Distance from Average Past Profile
Asymmetric Weighted Minkosky Distance
Fraud Risk Score
RUNTIME PHASE

Figure 2. *FraudBuster* approach: *time-windows-based* profile (light gray) and *time-series-based* profile (dark Gray) that are built for each user.

5.1 Training Phase

The training phase takes as input the list of historical transaction and builds, for each user, the profiles that model the spending pattern. As depicted in Fig. 2, given the inputs, *FraudBuster* performs the training in two steps: *recurrent spending pattern analysis* and *temporal profile construction*. This phase is repeated at the end of each time windows (e.g., monthly) to keep into consideration shifts and updates of the user's spending pattern.

Recurrent Spending Pattern Analysis. Transactions time series are analyzed to extract user's recurrent spending pattern following the procedure described in § 4. *FraudBuster* first applies the autocorrelation with the Bartlett filter to extract the autocorrelation coefficient. Then, it distinguishes periodic (i.e., monthly spending pattern) from non-periodic (i.e., non-monthly spending pattern) users by means of a threshold: if no autocorrelation coefficient overcomes this value, the user is labeled as non-periodic.

Temporal Profiles Construction. For each user, *FraudBuster* generates two classes of temporal profiles: the *time-series-based* and the *time-windows-based* profile. Each type of profile extracts different statistical features from the transaction's attributes (see Tab. 2), according to the type of model built.

The *time-series-based* profile is designed for periodic users only and evaluates the deviation of the current user's time series leveraging on their recurrent spending patterns observed in the past. In other words, it compares transactions time series in different time-periods and, therefore, it cannot be applied to non-periodic users since they don't show "similar" patterns that can be compared. Hence, it is built only on attributes that allows an accurate temporal analysis and for which the *time-windows-based* profile is not sufficient to discriminate anomalous transactions from legitimate ones: *Number of Transactions*, *Amount*, and *Total Amount* attributes.

The *time-windows-based* profile is designed for both users with periodic spending pattern and users with non-periodic spending pattern. It evaluates the deviation of the current user's behavior from the trained one aggregating transactions according to time windows of monthly length in a set of thresholds and histograms. In particular, it considers all features presented in Table 2 and does not compare "patterns" in data but only the aggregated attributes values in the time-windows. For these reasons, it is complementary to the *time-series-based* approach and is applied to both classes of users.

For both profiles, *FraudBuster* computes *the average past profile* and *the deviation* of past profiles from the average one. *FraudBuster* builds the *time-windows-based* profile by means of the average histograms for categorical attributes and thresholds for numerical ones. The thresholds are built based on the mean and the standard deviation of attribute values. The deviation from the average histogram is computed using the asymmetric weighted Minkowski distance [23]. For the *time-series-based* profile, *FraudBuster* uses the DTW Barycenter Averaging (DBA) [24] method to compute the average time series and the dynamic time warping (DTW) distance [4] for computing the deviation.

5.2 Runtime Phase

At runtime, *FraudBuster* evaluates nearly real-time each new transaction against the trained profiles and ranks users according to the aggregated risk of being defrauded. In particular, it measures the deviation of the current spending behavior with respect to the average profile built during the training phase.

Temporal Profile Scoring. For both profiles, it computes the current temporal profiles and its distance from the trained average one. For the *time-windows-based* profile, *FraudBuster* incrementally builds the current threshold and histogram distribution, while for the *time-series-based* profile, *FraudBuster* incrementally builds the current time series. Then, it computes the distance from the average profile with the asymmetric weighted Minkowski distance for histograms, the percentage gap for thresholds, and the dynamic time warping (DTW) distance for time series. Finally, the anomaly score is defined as the relative difference between the deviation computed during training and the current distance. By doing this, it considers the variance of the user's spending behavior. The final output is represented by the ranked list of users ordered by the aggregated risk score, which keeps into consideration the *time-windows-based* and the *time-series-based* anomaly score of each transaction.

6 System Details

In this section, we describe how *FraudBuster* works, giving a detailed description of the *time-windows-based* and the *time-series-based* profiles.

6.1 Time-windows-based Profile

For each numerical attribute, during the training phase, *FraudBuster* builds a threshold on the average plus the standard deviation of attribute value. At runtime, the system computes the anomaly score as the percentage gap between the trained threshold and the current values of incoming transactions.

For each categorical attribute, during the training phase, *FraudBuster* computes the average histogram distribution that counts the occurrences of each attribute value over each time windows and the mean deviation of historical user's spending pattern (i.e., histogram) from the average one. The distance is computed with the asymmetric weighted Minkowski distance [23], using a L-1 norm that linearly considers deviations between histograms. For each bin of the histogram we assign a weight based on the normalized frequency of the attribute that gives more importance to previously unseen values. At runtime, *FraudBuster* incrementally builds the current histogram distribution processing incoming transactions and computes the current distance from the average histogram built during the training phase.

The anomaly score is computed for each user, evaluating incoming transactions, and is defined as the relative difference between the current distance and the deviation computed during the training phase. *Time-windows-based* profile training and runtime phases are shown in Alg. 1.

Algorithm 1: Time-windows-based profile training and runtime phases (for user U).

```
Input: Historical transactions of user $U$ ;                                  // Training Phase
for $a$ in attributes $A$ do
    if $a$ in numerical then
        for $w$ in historical time windows $W$ do
            $past\_profile_w(a) = \sum_{t \in w} a(t)$ ;   // cumulative value of the attribute $a$
        end
        $avg\_profile(a) = \sum_{w \in W} past\_profile_w(a)/W$ ;              // average value
        $std\_dev(a) = \sqrt{(1/W)\sum_{w \in W}(past\_profile_w(a) - avg\_profile(a))^2}$ ;
         // deviation
        $threshold(a) = avg\_profile(a) + std\_dev(a)$
    end
    if $a$ in categorical then
        for $w$ in historical time windows $W$ do
            $past\_hist_w(a) = hist(a(t)|t \in w)$ ;         // histogram of the attribute c
        end
        $avg\_hist(a) = \sum_{w \in W} past\_hist_w(a)/W$ ;           // average past histogram
        $avg\_dev(a) = \sum_{w \in W} minkowski\_dist(past\_hist(a)_w - avg\_hist(a))/W$
    end
end
Result: $time\_win\_profile(U) = ([threshold(a)], [avg\_dev(a)])$ for $a$ in $A$
Input: Current transactions of user $U$ ;                                      // Runtime Phase
for $a$ in attributes $A$ do
    if $a$ is categorical then
        $curr\_profile(a) = \sum_{t \in curr\_time\_window} a(t)$ ;   // current cumulative value
        $time\_win\_score(a) = (curr\_profile(a) - threshold(a))/threshold(a)$;
    end
    if $a$ is numerical then
        $curr\_hist(a) = hist(a(t)|t \in curr\_t\_win)$ ;                  // current histogram
        $curr\_dist(a) = minkowski\_dist(curr\_hist(a) - avg\_hist(a))$;        // distance
        $time\_win\_score(a) = (curr\_dist(a) - avg\_dev(a))/avg\_dev(a)$;
    end
end
Result: $time\_win\_score(U) = \sum_{a \in A} t\_win\_score(a)$, if $t\_win\_score(a) > 0$
```

6.2 Time-series-based Profile

During the training phase, for each user, *FraudBuster* builds the average time series using the DTW Barycenter Averaging (DBA) [24] algorithm and the average deviation from the historical time series computing the DTW distance. At runtime, *FraudBuster* incrementally builds a new time series based on incoming transactions and computes the current dynamic time warping (DTW) distance from the average time series computed during the training phase. The anomaly score is computed for each user evaluating incoming transactions and is defined as the relative difference between the current distance and the average one. *Time-series-based* profile training and runtime phases are shown in Alg. 2.

Time series Comparison and Alignment. In order to compute the distance value among time series, it is necessary to *align* and *compare* them. To perform both comparison and alignment, we adopt the DTW algorithm [4] that computes the optimal alignment (i.e., match) and measures the similarity (i.e., distance) between the two time series. The sequences are "warped" non-linearly in the time dimension to determine a measure of their similarity. More formally,

Algorithm 2: Time-series-based profile training and runtime phases for user U.

```
   Input: Historical transactions of user $U$ ;                    // Training Phase
1  for $x$ in temporal attributes $X$ do
2      for period $\tau$ detected in the historical timespan $T$ do
3          $t\_series_\tau(x) = \{x(t)|t \in \tau\}$ ;                // past time series
4      end
5      $avg\_time\_series(x) = DBA(t\_series_\tau(x))$ ;        // average time series(x)
6      $avg\_dev(x) = \sum_{\tau \in T} DTW(t\_series_\tau(x), avg\_time\_series(x))/T$ ;     // average
        deviation
7  end
   Result: $time\text{-}series\text{-}based\_profile(U) = [avg\_dev(x)$ for $x$ in $X$ ]
   Input: Current transactions of user $U$ ;                       // Runtime Phase
8  for $x$ in temporal attributes $X$ do
9      $current\_time\_series_{\tau'}(x) = \{x(t)|t \in \tau'\}$ ;        // current time series
10     $curr\_dist(x) = DTW(current\_time\_series_{\tau'}(x), avg\_time\_series(x))$ ;
        // distance
11     $time\_series\_score(x) = (curr\_dist(x) - avg\_dev(x))/avg\_dev(x)$
12 end
   Result: $time\_series\_score(U) = \sum_{x \in X} time\_series\_score(x)$, if
           $time\_series\_score(x) > 0$
```

the DTW algorithm finds an optimal alignment between two ordered sequences, $X = [x_1, x_2, ..., x_n]$ of length N and $Y = [y_1, y_2, ..., y_m]$ of length M using a local distance metric $d(X, Y)$ between each pair of events x, y. As local distance we use a L-1 norm and for each pair of elements we obtain the cost matrix $C \in \mathbb{R}^{N \times M}$ defined as $C(n, m) = d(x_n, y_m)$. The alignment is a sequence $p = (p_1, p_2, ..., p_L)$ indexing over the pair (n, m) of C under the following constraints: 1) *boundary condition*, $p_1 = (1, 1)$ and $p_L = (N, M)$, requires that the first and last components of the aligned sequence should coincide; 2) *monotonicity condition* ($n_1 \leq n_2 \leq ... \leq n_L$ and $m_1 \leq m_2 \leq ... \leq m_L$) implies that the ordering must be preserved; 3) *step size condition* describes the relation between the original sequences and the final alignment: $p_{l+1} - p_l \in \{(1, 0), (0, 1), (1, 1)\}$ for $l \in [1 : L-1]$. As a consequence, the cost $c_p(X, Y)$ of an alignment p between X and Y with respect a local distance measure d is $c_p(X, Y) = \sum_{l=1}^{L} c(x_{n_l}, y_{m_l})$. The **optimal distance** $DTW(X, Y)$ between X and Y is obtained by minimizing $c_p(X, Y)$. In addition, we set additional constraints to avoid a wrong time series alignment and comparison. The *first problem* is that time distant elements can be aligned. To solve this, we introduce a global constraint (*Sakoe-Chiba band* region [4]), which considers an alignment admissible if the mutual time-distance between two values is bounded by a specific time value. The *second problem* is that each entry of the time series can be aligned to more than one entry of the second time series, making *FraudBuster* less resistant to fraudsters that try to simulate user behavior. Hence, we introduce a further constraint in the alignment region not allowing multiple alignment. Finally, the *Boundary condition* can introduce artifacts that are avoided by adding zero elements at the begin and the end of each sequence.

Time Series Averaging. As shown in Alg. 3 and Fig. 3, the average time series is constructed by aligning the set of time series with the DTW algorithm

Algorithm 3: DTW Barycenter Averaging (DBA).

```
// $i$ = { i-th historical time series }, $k$ = { k-th time series's component }
Input: Time series $t\_series_i = \{x_i(t_{i,j}) | t \in \tau, 1 \le i \le N, 1 \le j \le K\}$
1 $avg\_t\_series = \{x(t_{avg,j}) | t_{avg,j} \in \tau\ detected\}$ ;      // average time series definition
2 for $j$ in $K$ do
3     $t_{avg,j} = \sum_i^N barycenter\_time_j(t\_series_i) = \sum_i^N t_{i,j}/N$ ;    // Barycenter in time
4     $x(t_{avg,j}) = \sum_i^N barycenter\_value_j(t\_series_i) = \sum_i^N x_i(t_{i,j})/N$ ;    // Barycenter
      value
5 end
Result: $avg\_t\_series = \sum_{i=1}^N barycenter_{value,time}(DTW_{align}(t\_series_i))$
```

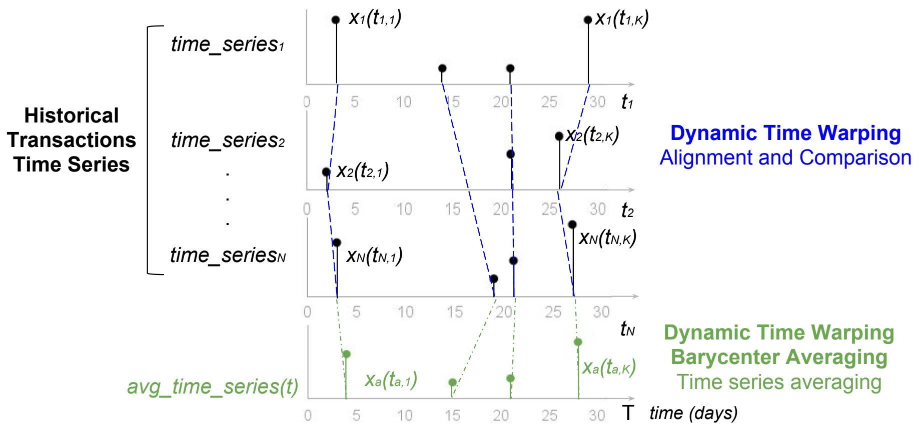


Figure 3. Average time series computation: dynamic time warping (DTW) alignment method (blue) and DTW Barycenter Averaging (DBA) (green).

and computing the barycenter of the attributes values. The algorithm starts aligning two time-series and computing a first “temporary” average time series by considering the barycenter between each of their component (i.e., in time and of the value). Then, it iteratively works in two steps until convergence. In the first step, it applies the DTW algorithm between each individual time series and the “temporary” average time series to find the optimal alignment. In the second step, it updates each element of the “temporary” average time series by computing the “barycenter” in time and of the value of each time series.

6.3 Temporal Profile Scoring

The anomaly scores can be simplified as $time\_windows\_score = \frac{(curr\_val - dev\_thr)}{dev\_thr}$ and $time\_series\_score = \frac{avg\_dist - avg\_dev}{avg\_dev}$, where $curr\_val$ is the feature's value for numerical variable or a distance's value between histogram for categorical variable, avg_dist is the distance between the average time series used as reference and the current one. To mitigate the problem of “divide by zero” that would have led to very high anomaly score even for small shift in user's behavior we add an arbitrary small smoothing parameter. Hence, the final scores are $time\_window\_score = \frac{curr\_\_val - dev\_\_thr}{dev\_\_thr + s}$ and $time\_series\_score = \frac{avg\_\_dist - avg\_\_dev}{avg\_\_dev + s}$.

Final Score Aggregation. Finally, to combine the two temporal profiles outputs in a risk of fraud score, we compute a weighted sum of the *time-series-based* and *time-windows-based* scores. In particular, to make the score less biased from features characterized by high mean and standard deviation we first apply a z-score standardization: $z\_score_{u,f} = \frac{score_{u,f} - mean_f}{std\_dev_f}$, where $mean_f$ and std_dev_f are the mean and standard deviation of scores of the attribute f. We estimate *FraudBuster* parameters by choosing the values that maximizes detection performances. To avoid over-fitting, we conduct this parameter tuning on a subset of the overall dataset (about 3,000 randomly sampled users with at least one transaction per month, to have a more stable dataset).

7 Experimental Evaluation

The goal of this experimental evaluation is to measure the effectiveness of *FraudBuster* in detecting *salami-slicing* frauds, *i)* comparing the results with the temporal profile of [9], *ii)* proving the effectiveness of temporal pattern exploitation in the detection of these frauds, *iii)* assessing the robustness of *FraudBuster* against mimicry-attacks, and *iv)* evaluating the capability to detect real-world *salami-slicing* frauds. It is important to highlight that for all these experiments we focus on the detection of *salami-slicing* frauds only, since the goal of this work is to provide a mitigation to this insidious menace.

7.1 Dataset Description

The cooperation with an Italian banking group gave us the opportunity to test our system on the real-world *unlabeled* dataset analyzed in [9]. The dataset, summarized in Tab. 2, contains the fully anonymized record of the transactions performed through the Internet banking services of one of the largest Italian banking groups (i.e., bank transfers, phone recharges and prepaid debit cards) and spans five months, from April to August 2013. As confirmed by our collaborators, it contains no known frauds (i.e., the dataset was preprocessed removing anomalous instances). This dataset allowed us to evaluate the detection power of *FraudBuster* against fraudulent scenarios, to prove the effectiveness of temporal pattern exploitation, and to compare the results with the temporal profile of [9].

Table 2. Number of transactions, number of users, and selected features of the dataset.

Dataset	**Transactions**	**Users**	**Selected Features**
Bank Transfer	379,242	47,909	# of Transaction, Total Amount, # of New IBAN , IBAN, IBAN_CC, ASN_CC, Amount(Discretized)
Phone Recharge	50,708	15,683	# of Transaction, Total Amount, # of New Phone Number, Phone Number, ASN_CC, Amount(Discretized)
Prepaid Debit Card	34,583	8,424	# of Transaction , Total Amount, # of New CardID, CardID, ASN_CC, Amount(Discretized)

We immediately noticed that the transactions are characterized by a "weekly" pattern of five days (working days) with a high value of records, followed by two days (weekend) of lower activity. We also noticed that August has a lower volume of transactions, likely due to the typical Italian summer holiday period.

Dataset Features. Beyond the obvious features (e.g., amount and timestamp), the dataset contains the following attributes: **UserID** that is a unique ID associated to the "author" of the transaction; **IBAN** and **IBAN_CC** of the transaction recipient and its country code (CC); **CardID**, a unique ID associated to each prepaid debit card; **ASN_CC**, the Autonomous System Number (ASN) and its CC from which the user was connected when issuing the transaction. We purposely ignore IP due to its high variance (e.g., dynamic IP address, use of proxy). Personally-Identifiable Information (PII) (e.g., UserID, IBAN) is hashed.

7.2 Evaluation Approach and Metrics

The evaluation of *FraudBuster* is particularly challenging due to the lack of a ground truth. Hence, leveraging on the domain expert's knowledge and on the analysis of some real *salami-slicing* frauds attack schemes, we reproduced synthetically-generated frauds that replicate real attacks performed against online banking users. Following the threat model described in § 2, we assume that the goal of the attacker is to perform frauds while remaining undetected. Therefore, frauds are characterized by low and medium daily amounts, are executed during working hours either from the victim's device that is used as a proxy or from unknown devices (i.e., foreign ASN_CC). For the bank transfer context, we test both the case of national and foreign IBAN_CC. In Tab. 3 we list all our threat scenarios. We put effort to produce synthetic transactions that resemble the real ones to be as realistic as possible and to avoid the overfitting of our approach to "trivial" fraudulent transactions. For this reason, features' distributions are extracted on the basis of an in-depth dataset analysis to make them "indistinguishable" from genuine transactions. The representativeness of these scenarios was later confirmed by the real-world frauds analyzed in Exp. 4.

We split the dataset following the *holdout* method and using three months for building the profiles and the last month (plus synthetic injected transactions) for the detection performance analysis. Even if *FraudBuster* works near real-time, we decide to evaluate the detection at the end of the month to fairly compare the performance metrics over the same period. As explained in § 6, *FraudBuster* computes, for each incoming transaction, the anomaly score on the basis of the learned model and assigns an aggregated risk score to each user. Finally, it ranks the pair $< user, score >$ in descending order, to support the analysts' ex-post analysis (i.e., manual investigation of fraudulent transactions), by making them investigate only on the top N users of the ranking and on transactions that deviates most from the user's spending pattern. For the first four experiments, after training, we randomly select N users and inject (blindly to *FraudBuster*) the synthetically-generated *salami-slicing* frauds into their transactions belonging to the testing data. Then, we use *FraudBuster* to analyze the testing data

Table 3. Fraud scenarios. The label "Equally" means "frauds are executed toward different accounts with repetitions", "Max one" means "maximum one fraud per account", "All to one" means "all frauds are executed toward one account".

Fraud scenario	#Frauds	IBAN	ASN	Beneficiary Distribution	Amount (€)		
					Bank Transfer	Prepaid card	Phone recharge
1	10	National	National	Equally	100-500	50-100	5-25
2	10	National	National	Equally	500-1,500	100-250	25-50
3	10	National	National	Equally	1,500-3,000	250-500	50-100
4	10	National	National	**All to One**	500-1,500	100-250	25-50
5	10	National	National	**Max one**	500-1,500	100-250	25-50
6	5	National	National	Equally	500-1,500	100-250	25-50
7	5	**Foreign**	National	Equally	500-1,500	-	-
8	5	National	**Foreign**	Equally	500-1,500	100-250	25-50
9	5	National	National	Equally	2,500-7,500	500-1000	100-200

and to rank users. To evaluate the detection performance, we consider the top N users in the ranking. The value of N is chosen according to bank workforce, but from our information, the team of analysts is able to analyze around 1-5% of users. Therefore, in these experiments, we consider the top $N = 1\%$ users of the final ranking from those ranked as anomalous. We perform these operations for each threat scenario described in Tab. 3, which specifies the number, the transaction's beneficiary distribution, and the features' values of frauds injected per user. Moreover, we repeat the test 50 times and average the results to avoid statistical artifacts due to the injection pattern of frauds to random users.

Under these assumptions and considering as defrauded the top N users in the ranking, a True Positive (TP) is a defrauded user correctly ranked as defrauded, False Positive (FP) is a legitimate user wrongly ranked as defrauded, a False Negative (FN) is a defrauded user wrongly ranked as legitimate, and a True Negative (TN) is a legitimate user correctly ranked as non-anomalous. Then, we compute the well-known evaluation metrics of True Positive Rate (TPR), which compute the percentage of correctly identified defrauded users and False Positive Rate (FPR), which compute the percentage of legitimate users who are wrongly identified as defrauded: $TPR = \frac{TP}{TP+FP}$, $FPR = \frac{FP}{TN+FP}$. We also compute the Average Precision (AP) that summarizes the precision-recall curve as the weighted mean of the precision achieved at every position in the ranking. By doing this, it takes into account the position (i.e., the order) of defrauded users in the ranking: $AP = \frac{\sum_{k=1}^{n} P(k)\cdot F(k)}{R}$, where R is the number of defrauded users, N is the number of user considered in the raking, $P(k)$ is the precision at cut off k, and $F(k) = 1$ if the k^{th} user is fraudulent, 0 otherwise. Since our dataset is highly unbalanced in favor of normal users, we use the Average Accuracy (AA) metrics. The AA is an average of the accuracy obtained for both fraudulent and normal users classes: $AA = \frac{1}{2}\left[\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right]$. Finally, we graphically represents *FraudBuster* performances with the Receiver Operating Characteristic (ROC) that express the ratio between TPR and FPR.

It is important to highlight that the described evaluation approach and metrics, besides giving an index of the detection performance, allow us to indirectly evaluate *FraudBuster* from the point of view of the cost of challenging frauds and

Table 4. Experiment 1 results. Considered metrics: True Positive Rate (TPR), Average Accuracy (AA), and Average Precision (AP). The label *FB* refers to *FraudBuster*, while *BS* refers to temporal profile of *BankSealer*.

	Bank Transfer						Prepaid Debit Card						Phone Recharges					
	TPR		AP		AA		TPR		AP		AA		TPR		AP		AA	
Scenario	FB	BS	FB	BS	FB	BS	FB	BS	FB	BS	FB	BS	FB	BS	FB	BS	FB	BS
1	**69**	15	**61**	12	**84**	57	**90**	44	**95**	33	**95**	71	**97**	70	**99**	78	**98**	85
2	**75**	28	**70**	21	**87**	63	**91**	52	**95**	46	**95**	75	**97**	80	**99**	90	**99**	90
3	**79**	40	**77**	34	**89**	70	**93**	60	**97**	62	**96**	80	**98**	89	**99**	96	**99**	94
4	**82**	28	**80**	21	**91**	63	**91**	52	**95**	46	**95**	75	**98**	80	**99**	90	**99**	90
5	**74**	28	**69**	21	**87**	63	**95**	52	**98**	46	**97**	75	**98**	80	**99**	90	**99**	90
6	**42**	16	**30**	11	**70**	57	**75**	31	**66**	23	**87**	65	**87**	61	**93**	67	**93**	80
7	**79**	16	**73**	11	**89**	57	-	-	-	-	-	-	-	-	-	-	-	-
8	**76**	16	**79**	11	**88**	57	**88**	31	**90**	23	**94**	65	**98**	61	**99**	67	**99**	80
9	**62**	16	**51**	11	**81**	57	**82**	55	**76**	77	**91**	55	**98**	87	**99**	93	**99**	95
mixture	**70**	25	**65**	19	**85**	62	**85**	47	**92**	41	**92**	73	**96**	76	**98**	85	**99**	88

amount of funds protected from defrauding attempts. In fact, while the cost of a FP is the time spent by the analyst in the verification process, the cost of a FN is the stolen amount and the loss of trust in the financial institution. Hence, a high TPR, AA, and AP associated to a low FPR guarantees that *FraudBuster* is correctly ranking frauds while reducing the rate of "false" alarms, which directly impacts the banking analyst activity, and the amount of founds stolen (i.e., TN). Finally, by limiting the analysis to the top $N\%$ positions in the ranking, we are putting a cap on the costs of challenging frauds (e.g., banking analysts).

7.3 Experiment 1: Evaluation Against Fraud Scenarios

In this experiment, we show the effectiveness of *FraudBuster* comparing the results with the temporal profile of BankSealer [9] only, since it detects "stealthy frauds" repeated in times, like *salami-slicing* frauds. For each test, we inject the synthetic frauds described above and summarized in Tab. 3, equally distributed in the timespan of the testing data. The classification performance is shown in Tab. 4, while the ROC curve for bank transfers is presented in Fig. 4. We show the detection performance for the bank transfers context only for brevity, but similar results were obtained for the other contexts. Remarkably, Fig. 4a shows that *FraudBuster* outperforms the temporal profile of BankSealer in all fraud scenarios under analysis. In particular, *FraudBuster* reaches a detection rate of 82%, 95%, and 98%, an average precision of 80%, 98%, and 99% and an average accuracy of 91%, 97%, and 99% in the bank transfer, phone recharges, and prepaid cards dataset respectively. These results represent an improvement in all the metrics under analysis that is 30-60% larger with respect to the temporal profile of [9]. Therefore, *FraudBuster* ranks *salami-slicing* frauds in higher position with respect to [9]. ROC analysis shown in Fig. 4c and Fig. 4e, highlights that *FraudBuster* maintains good performance in almost every threat scenario. The best performance is obtained in case of 10 frauds injected per users, considering national beneficiary and low amounts, with a TPR up to 82% for the bank transfers context (95% and 97% for phone recharges and debit card contexts).

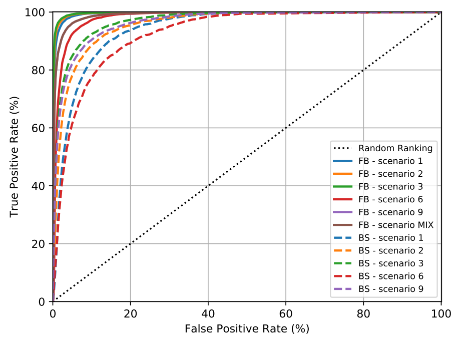
Moreover, the Average Accuracy (AA) is always particularly high, meaning that *FraudBuster* successfully reduces the number of false positives while maintaining a good TPR. The lowest performance is obtained in scenario 6, since it represents the most difficult to detect due to the low number of frauds and their high resemblance to legitimate transactions. *FraudBuster* shows a TPR of 42% in the bank transfer context (75% and 82% in the prepaid debit cards and phone recharges contexts). However, also in this case *FraudBuster* detects 30% more *salami-slicing* frauds than BankSealer and puts all fraudulent users in the top 3% of the ranking (see Fig. 4e). The results obtained on prepaid debit card and phone recharges are, on average, better with respect to bank transfer domain, since in these contexts frauds toward previously unseen values are easier to spot.

7.4 Experiment 2: Effectiveness of the Time-series-based Profile

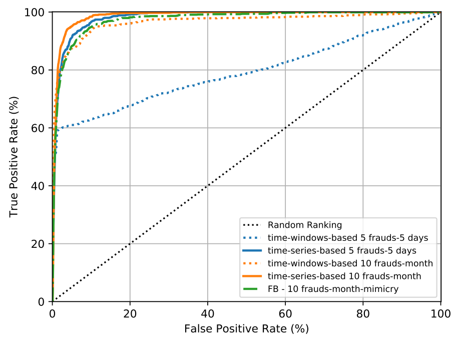
The purpose of this experiment is to compare the detection rate of the *time-series-based* profile with respect to the *time-windows-based* profile alone for each attribute. Since the *time-series-based* profile is defined only for periodic users, here we design the experiment following the same methodology of the previous one, but keeping only periodic users. We consider two different scenarios: (1) 5 random frauds are injected per infected user over 5 days; (2) 10 random frauds are injected per infected user over the whole month. Fig. 4 shows that the *time-series-based* profile and the *time-windows-based* profile have similar performances in the first part of the ROC curves but, after few FP cases, the detection rate of the *time-series-based* profile outperform the one of the *time-windows-based* profile by 20-30%. This is even more evident in the testing scenario with only 5 frauds where both profiles detect “easy-to-spot” frauds, but only the *time-series-based* one, which exploit recurrent temporal patterns, is able to catch “stealthiest” ones. These results highlight the effectiveness of the *time-series-based* profile in the detection of *salami-slicing* frauds.

7.5 Experiment 3: Evaluation Against Mimicry Attack

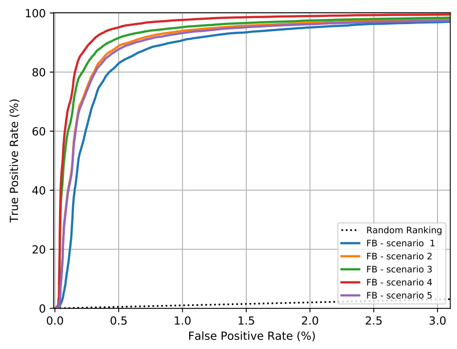
Though there has been a good amount of research on fraud analysis, the security of these systems against evasion attacks seems not to have received much attention in the banking fraud analysis context. There are many papers proposing new techniques for fraud detection, and authors often try to measure their detection power by testing whether they can detect currently-popular attacks. However, the notion of security against adaptive adversarial attacks, defined as “mimicry attacks” [25], is much harder to measure. In this experiment we investigate the performance of *FraudBuster* under the assumptions that the fraudster knows the detection algorithm and the spending habits of the victim. With this data the attacker can approximate the behavior of the user in order to generate fraudulent transactions to trick the anti-fraud framework and silently commit frauds. Therefore, in this scenario, we inject 10 frauds mimicking user’s behavior (i.e., we follow the target user’s spending pattern). The results of this experiment are



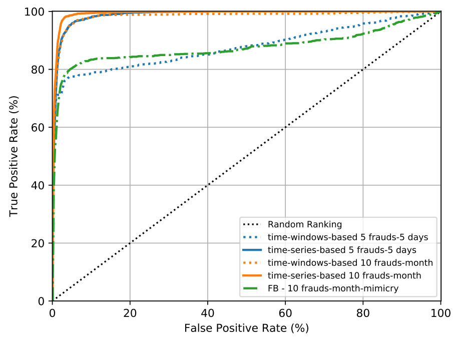
(a) **Exp. 1:** *FraudBuster* vs BankSealer



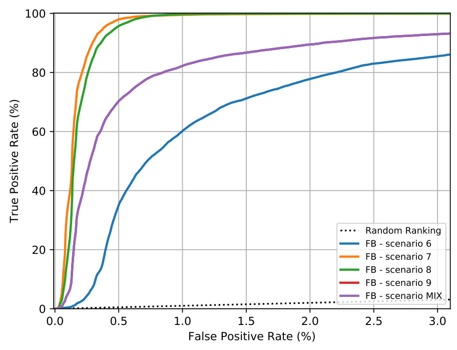
(b) **Exp. 2 and 3:** Number of transactions



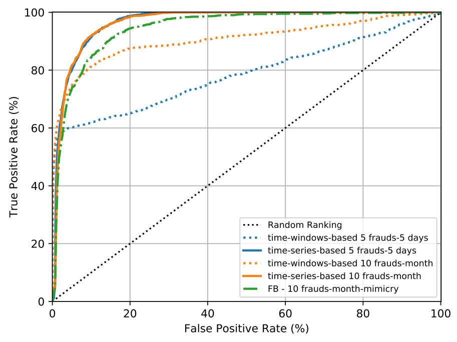
(c) **Exp. 1:** 10 frauds injected.



(d) **Exp. 2 and 3:** Histogram amount.



(e) **Exp. 1:** 5 frauds injected.



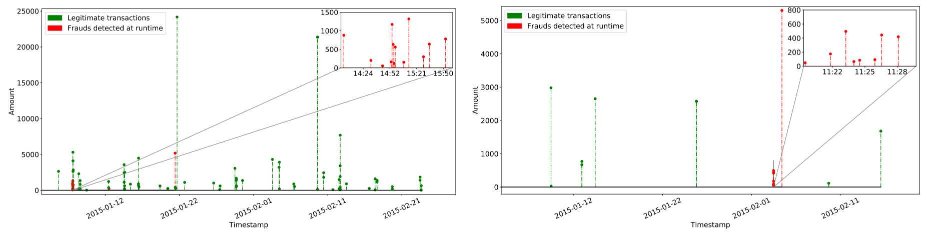
(f) **Exp. 2 and 3:** Total Amount.

Figure 4. Experiment 1, 2, and 3 Receiver Operating Characteristic (ROC) curves, varying the number N of users considered as defrauded. *FB* stands for *FraudBuster*, *BS* stands for *BankSealer*. Fig. 4c and Fig. 4e show an enlargement of the ROC curves.

superimposed in Fig. 4 to have a better comparison with *FraudBuster*'s performance. As expected, *FraudBuster* performs worse than the previous experiments since frauds are more similar to legitimate ones and, hence, more difficult to detect. In fact, the mimicry-attack tries to keep a lower risk score, spreading frauds in the ranking. However, it is important to highlight that *FraudBuster* is able to mitigate the mimicry-attack keeping an overall detection rate around 65% and pushing frauds in the top 3% of the ranking. This allows to detect at least the majority of frauds and stop them. The good results obtained are mainly due to the combination of the *time-windows-based* and the *time-series-based* profiles, which require more effort to an attacker that wants to mimicry users' behaviors.

7.6 Experiment 4: Evaluation Against Real-world Frauds

Due to the good results obtained in the previous experiments, we deployed *FraudBuster* in the real-world setting of a large national banking group and measured its detection performance. *FraudBuster* analyzed 482,930 bank transfer transactions executed by 58,562 users from October 2014 to February 2015. In particular, similarly to previous experiments, we build the spending profiles on the first two months and analyze the performance on the sequent months, progressively updating the temporal models with legitimate transactions only. In order to have a ground-truth, we ask banking analysts to manually inspect and label frauds in the ranking. *FraudBuster* was able to detect *salami-slicing* frauds, not detected by the other protection system installed, for a total of almost 20 fraudulent transactions. Tab. 5 contains two different examples of the real frauds detected by *FraudBuster*. The same fraudulent transactions are also graphically presented in Fig. 5, which show (with the red color) the amount of funds stolen in time. The fraudulent transactions in Fig. 5a and Tab. 5a present amount less than 1,500 € and represents an attacker that exploits different sessions (by hijacking real sessions or deceiving the victims to insert session tokens) to perform his/her frauds. In this case, frauds are executed toward foreign and Italian bank accounts, which, in some cases, are also used twice. These characteristics make these *salami-slicing* frauds as a combination of scenario 4 (single



(a) (b)

Figure 5. Experiment 4. Examples of real frauds detected by *FraudBuster*. Legitimate transactions are represented in green, while frauds in red.

IP	SID	Timestamp	Amount	IBAN
1..8	1..7	2015-01-07 14:03:06	880.17	GB..8
1..8	5..1	2015-01-07 14:32:17	200	IT..a
1..8	5..1	2015-01-07 14:44:52	55.18	IT..7
1..8	5..1	2015-01-07 14:54:08	159.2	IT..7
1..8	5..1	2015-01-07 14:55:06	1,171.2	IT..5
1..8	5..1	2015-01-07 14:56:15	631.96	IT..d
1..8	5..1	2015-01-07 14:57:16	120	IT..a
1..8	5..1	2015-01-07 14:58:29	561	IT..5
1..8	5..1	2015-01-07 15:07:43	150	FR..c
1..8	5..1	2015-01-07 15:13:11	1,318.82	IT..5
1..8	5..1	2015-01-07 15:29:01	300	IT..d
1..8	5..1	2015-01-07 15:35:17	640	IT..a
1..8	5..1	2015-01-07 15:52:50	780	FR..5
a..5	0..0	2015-01-12 12:01:23	224.74	GB..8
1..1	1..d	2015-01-21 10:07:22	5,187.5	FR..f

(a)

IP	SID	Timestamp	Amount	IBAN
2..7	5..c	2015-02-03 11:20:08	48	IT..a
2..7	5..c	2015-02-03 11:22:23	175	IT..4
2..7	5..c	2015-02-03 11:23:45	495	GB..6
2..7	5..c	2015-02-03 11:24:28	66	IT..2
2..7	5..c	2015-02-03 11:24:59	84	IT..3
2..7	5..c	2015-02-03 11:26:20	93	IT..b
2..7	5..c	2015-02-03 11:26:56	443.52	IT..1
2..7	5..c	2015-02-03 11:28:25	418.88	IT..a
3..a	G..5	2015-02-04 9:52:10	5,300.6	GB..4

(b)

Table 5. Experiment 4. Examples of real frauds detected by *FraudBuster*. *IP*, *SessionID* (SID), and *IBAN* are hidden for privacy reason.

IBAN destination), scenario 7 (foreign IBAN destination), and 9 (medium-high amount) (Tab. 3). The fraudulent transactions in Fig. 5b and Tab. 5b present low amounts and are mainly in the same day, except for the last fraud that try to steal a medium-high amount. These characteristics are similar to the ones of injection scenario 5 (small amount, different IBAN destination), summarized in Table 3. Besides *salami-slicing* frauds, *FraudBuster* was able to improve the detection performance of defrauded users by placing in the first 1% of the ranking 8 defrauded users not detected by the already installed detection system.

8 Conclusions

In this paper, we provided an in-depth temporal analysis on real data exploiting time series techniques to discern recurrent vs. non-recurrent spending patterns, which is, to the best of our knowledge, the first of such analysis in the literature.

We presented *FraudBuster*, a framework that exploits users' recurrent vs. non-recurrent spending patterns to detect *salami-slicing* frauds. *FraudBuster* automatically extracts end user's spending pattern and evaluates the deviation from its historical one. *FraudBuster* builds two models based on user's periodicity. The first model is the *time-series-based* profile that is designed for users with a periodic spending pattern only. The second model is the *time-windows-based* profile that is designed for all users independently from the periodicity and aggregates user's transactions according to time-windows. For each incoming transaction, *FraudBuster* measures the deviation (i.e., anomaly score) of the user's spending activity from the learned models. The final output is an aggregated score that quantifies the risk of a user of being defrauded. By doing this, *FraudBuster* supports the necessary ex-post analysis (i.e., manual investigation of frauds), making analysts focusing only on highly ranked users and on transactions that deviates most from the user's spending pattern.

We tested *FraudBuster* in the real-world context of a large national bank. Leveraging the domain expert's knowledge, we reproduced *salami-slicing* frauds (in a controlled environment) performed against banking users, and recorded the resulting fraudulent transactions. *FraudBuster* outperformed the state-of-the-art temporal approach considered by detecting up to 60% more defrauded users. In particular, *FraudBuster* was able to reach a detection rate up to 82%, 95%, and 99% respectively in the *bank transfers*, in the *prepaid debit cards*, and in the *phone recharges* context. In addition, we demonstrated the benefits of the time-series analysis and we investigated the performance of *FraudBuster* against mimicry attacks and real-world *salami-slicing* frauds. *FraudBuster* was able to identify real-world *salami-slicing* frauds, for a total of almost 20 fraudulent transactions.

The main limitation of this work is related to the scarcity of data. An analysis of a larger dataset could have revealed more information about user's spending pattern. A future work could be the application of the Matching Pursuit decomposition [26] algorithm that allows to decompose a time series in a linear combination of patterns, hence, detecting different mixture of periodicities. Moreover, to include a more precise prediction of user's activity in windows-based profiles, a numerical prediction based on ARMA process can be implemented. In addition, it was not possible to exploit the semantic data (e.g., name of the recipient, payment description/reason) due to regulatory and privacy reasons. A future extension could be the extraction of privacy-preserving semantic features from the bank side. Finally, the results of the spending pattern analysis of § 4 heavily depend on the modeled country (i.e., what follows a monthly pattern in Italy could follow a weekly pattern in a different nation, or even no pattern at all). Therefore, the findings are not easy to generalize. However, the described methodology is independent from the context and the recurrent spending patterns analysis and classification can applied automatically on any dataset.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme, under grant agreement No.700326 (RAMSES project).

References

1. Wei, W., Li, J., Cao, L., Ou, Y., Chen, J.: Effective detection of sophisticated online banking fraud on extremely imbalanced data. World Wide Web **16**(4) (July 2013) 449–475
2. Emm, D., Unuchek, R., Kruglov, K.: Kaspersky Security Bulletin 2016. Technical report, Kaspersky Lab (2017)
3. Bidgoli, H.: Handbook of Information Security, Threats, Vulnerabilities, Prevention, Detection, and Management. Handbook of Information Security. Wiley (2006)
4. Müller, M.: Dynamic time warping. Information retrieval for music and motion (2007) 69–84
5. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Comput. Surv. **41**(3) (July 2009) 15:1–15:58

6. KasperskyLab: Banking Trojans: mobile's major cyberthreat (09 2015)
7. Phua, C., Alahakoon, D., Lee, V.: Minority report in fraud detection: classification of skewed data. SIGKDD Explor. Newsl. **6**(1) (June 2004) 50–59
8. Bolton, R.J., Hand, D.J.: Statistical fraud detection: A review. Statistical Science **17** (2002)
9. Carminati, M., Caron, R., Maggi, F., Epifani, I., Zanero, S.: BankSealer: A decision support system for online banking fraud analysis and investigation. Computers & Security **53** (2015) 175–186
10. Chen, L.M., Chen, M.C., Sun, Y.S., Liao, W.: Spectrum analysis for detecting slow-paced persistent activities in network security. In: ICC, IEEE (2013)
11. Aiello, M., Cambiaso, E., Mongelli, M., Papaleo, G.: An on-line intrusion detection approach to identify low-rate DoS attacks. Security Technology (ICCST), 2014 International Carnahan Conference on (2014)
12. Cover, T., Thomas, J.: Elements of information theory. Wiley, New York (1991)
13. Janacek, G.J., Bagnall, A.J., Powell, M.: A Likelihood Ratio Distance Measure for the Similarity Between the Fourier Transform of Time Series. In Ho, T.B., Cheung, D.W.L., Liu, H., eds.: PAKDD. Volume 3518 of Lecture Notes in Computer Science., Springer (2005) 737–743
14. Seyedhossein, L., Hashemi, M.: Mining information from credit card time series for timelier fraud detection. In: Telecommunications (IST), 2010 5th Intl. Symposium on. (2010) 619–624
15. Bhattacharyya, S., Jha, S., Tharakunnel, K., Westland, J.C.: Data mining for credit card fraud: A comparative study. Decision Support Systems **50**(3) (2011) 602–613
16. Krivko, M.: A hybrid model for plastic card fraud detection systems. Expert Syst. Appl. **37**(8) (2010) 6070–6076
17. Kundu, A., Panigrahi, S., Sural, S., Majumdar, A.K.: BLAST-SSAHA Hybridization for Credit Card Fraud Detection. IEEE Trans. Dependable Sec. Comput. **6**(4) (2009) 309–315
18. Altschul, S., Gish, W., Miller, W., Myers, E., Lipman, D.: Basic local alignment search tool. Journal of Molecular Biology **215** (1990) 403–410
19. van der Vaart, A.: TIME SERIES. Technical report, Vrije Universiteit Amsterdam (2010) pages 67–79.
20. Oppenheim, A.V., Schafer, R.W., Buck, J.R.: Discrete-Time Signal Processing. Prentice Hall, Upper Saddle River, NJ (1999)
21. Bartlett, M.: Periodogram Analysis and Continuous Spectra. Biometrika 37 (1950)
22. Cha, S.H., Srihari, S.N.: On measuring the distance between histograms. Pattern Recognition **35**(6) (2002) 1355–1370
23. Tran, N.M., Osipenko, M., Härdle, W.K.: Principal Component Analysis in an Asymmetric Norm. arXiv preprint arXiv:1401.3229 (2014)
24. Petitjean, F., Ketterlin, A., Gançarski, P.: A global averaging method for dynamic time warping, with applications to clustering. Pattern Recognition **44**(3) (2011) 678–693
25. Wagner, D., Soto, P.: Mimicry Attacks on Host-Based Intrusion Detection Systems. In: CCS02. (2002)
26. Mallat, S., Zhang, Z.: Matching pursuits with time-frequency dictionaries. Signal Processing, IEEE Transactions on **41**(12) (December 1993) 3397–3415