No Smurfs: Revealing Fraud Chains in Mobile Money Transfers

Maria Zhdanova*, Jürgen Repp*, Roland Rieke*†, Chrystel Gaber‡ and Baptiste Hemery§
*Fraunhofer Institute SIT, Darmstadt, Germany
email:{maria.zhdanova,juergen.repp,roland.rieke}@sit.fraunhofer.de
†Philipps-Universität Marburg, Marburg, Germany
‡Orange, Caen, France
§Normandie Université, Caen, France; UNICAEN, GREYC, F-14032 Caen, France;
ENSICAEN, GREYC, F-14032 Caen, France; CNRS, GREYC, F-14032 Caen, France
email: baptiste.hemery@ensicaen.fr

***Abstract*—Mobile Money Transfer (MMT) services provided by mobile network operators enable funds transfers made on mobile devices of end-users, using digital equivalent of cash (electronic money) without any bank accounts involved. MMT simplifies banking relationships and facilitates financial inclusion, and, therefore, is rapidly expanding all around the world, especially in developing countries. MMT systems are subject to the same controls as those required for financial institutions, including the detection of *Money Laundering (ML)* – a source of concern for MMT service providers. In this paper we focus on an often practiced ML technique known as micro-structuring of funds or smurfing and introduce a new method for detection of *fraud chains* in MMT systems. Whereas classical detection methods are based on machine learning and data mining, this work builds on Predictive Security Analysis at Runtime (PSA@R), a model-based approach for event-driven process analysis. We provide an extension to PSA@R which allows us to identify fraudsters in an MMT service monitoring network behavior of its end-users. We evaluate our method on simulated transaction logs, containing approximately 460,000 transactions for 10,000 end-users, and compare it with classical fraud detection approaches. With 99.81% precision and 90.18% recall, we achieve better recognition performance in comparison with the state of the art.**

***Keywords*-money laundering; mobile money transfer systems; fraud detection; predictive security analysis; process behavior analysis; machine learning**

I. Introduction

Mobile Money Transfer (MMT) services are financial services provided by a Mobile Network Operator (MNO) that enable transfers of funds (cash) between service subscribers through the use of mobile channels. These services do not imply a banking contract and are not tied to a bank account. Instead, the subscribers use mobile devices and a digital currency, a.k.a. electronic money, to commit transactions. MMT is a fast growing market expected to reach over 450 million subscribers in 2017, with a mobile transaction value of more than $721 billion, according to Gartner [1]. More than 150 services are currently deployed in 72 countries, for the most part, in Sub-Saharan Africa, where in 2012 there were twice as many users for MMT services than for Facebook [2]. Examples of the most successful deployments are M-PESA and Orange Money. As of 2012, M-Pesa, developed between Safaricom and Vodafone, has been used by 15 million people in Kenya [3]. Orange Money is deployed in 10 countries across the region and has about 4 million subscribers [4].

The growing financial market is an attractive target for attackers and fraudsters. MMT is an incentive for various types of fraud driven by different actors involved in the MMT ecosystem [5]. In particular, serious concerns have been raised regarding the risk of Money Laundering (ML) in MMT services, due to the provided ability for worldwide funds exchange in digital currencies coupled with the lack of oversight [6]. If proper controls are not deployed, fraudsters can get access to the service without disclosing their identity to the MNO, for example, taking advantage of prepaid phones, "pooling" and delegation of mobile devices [7]. Due to Anti-Money Laundering (AML) regulations in most countries, it is compulsory for MMT service providers to report ML activities. Therefore, ML detection is vitally important for MNOs to be able to run mobile financial services and prevent reputation risks. The goal of ML is to disguise the origin of illegal incomes and make them appear legitimate using a range of strategies to evade AML controls. One of the often practiced techniques is *smurfing* that involves multiple third parties, so-called "smurfs", conducting money transfers on behalf of fraudsters, so that the transaction amounts are kept below reporting levels [7], [8]. A smurf, or more commonly termed *money mule*, is recruited by fraudsters as a financial intermediary who accepts money from one fraudster and forwards it to another fraudster for a fee. Mules are often engaged through the use of phishing strategies, such as bogus jobs, and not aware that they are dragged into illegal activities [9].

The common approach to fraud detection in MMT is to use classical statistical methods such as machine learning and data mining [10]–[13]. For example, neural networks are used in industrial products by Visa [14] and M-Pesa [15]. However, these methods need a training database, which for ML can be difficult to obtain, and often produce results that are not easy to interpret. A serious challenge for ML detection is that fraudulent transactions may have parameters (e.g., amount, frequency) very close to regular money transfers and be nearly indistinguishable from the behavior of legitimate subscribers.

We propose an alternative method for ML detection in MMT services that is able to identify fraudsters and money mules engaged in a fraud chain. In contrast to the classical approaches, this method does not depend on any prior information about fraud patterns or samples of ML transactions, nevertheless, it shows comparable (or better) recognition performance in terms of the precision and recall metrics. At that, it allows analyst efficiency to be increased giving interpretable alerts. Our method for fraud chain detection builds on model-based approach for event-driven process analysis Predictive Security Analysis at Runtime (PSA@R) [16]. Essentially, the method introduced in this paper is an extension to PSA@R that allows us to define and synchronize processes on different abstraction levels using synthetic events derived from the runtime behavior of MMT users. As no public data on ML is available, we evaluate the proposed method on simulated transaction logs produced by an advanced MMT simulator based on a multi-agent platform [17]. We use the same logs for the comparative study of several machine learning algorithms in order to obtain comparable results required to judge about the efficiency of the new detection method. For evaluation purposes, we implement our method as a plug-in to process modeling and verification tool Predictive Security Analyzer (PSA) [18].

The remainder of this paper is organized as follows: Section II presents the MMT ecosystem and defines the fraud scenario. Section III discusses the state of the art in fraud detection, and Section IV formulates the design goals and introduces the alternative method for detection of fraud chains. Section V reports the protocol of our experiments and discusses the obtained results in comparison with machine learning algorithms. Finally, Section VI concludes the paper and outlines directions for future research.

II. Fraud Chains in Mobile Payments

A. MMT Ecosystem

An MMT service is a complex ecosystem that involves an MNO, a private bank, the country's Central Bank in which the service is deployed, and service subscribers. An MNO provides infrastructure and communication services and in partnership with a private bank emits electronic money called *mMoney* [19]. mMoney is a digital equivalent of funds (cash) that can be used solely within the service system to conduct mobile-enabled financial operations, such as Airtime Recharge (AR), Money Deposit (MD) and Money Withdrawal (MW), Merchant Payment (MP), domestic and international Client-to-Client transfer (C2C). End-users and retailers are service subscribers, holding a prepaid mobile account *mWallet* stored on an MMT platform. As shown in Fig. 1, in order to conduct C2C transfer, end-user Alice needs to convert her cash to mMoney and deposit its amount into her mWallet with the help of the retailer $R1$. Then, Alice can use her mobile device to transfer mMoney to Bob, if he is subscribed to the same MMT service. On receiving the transfer, Bob can withdraw cash from his mWallet at the retailer $R2$.

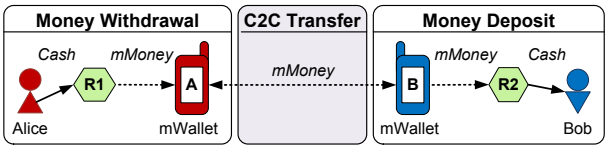


Figure 1. C2C funds transfer using MMT service.

B. Fraud Scenario

The fraud scenario studied in this paper originates from an often practiced ML technique known as micro-structuring of funds or *smurfing*. Smurfing involves multiple third parties or third party accounts that conduct money transfers on behalf of fraudsters so that large amounts of "dirty" money are distributed among a number of smaller transactions [8]. It allows fraudsters to hide ML activities from controllers and evade AML reporting requirements, thus, reducing the likelihood of fraud detection.

Definition 1 (Fraud chain): A *fraud chain* is a group of end-users of an MMT service identified by their mobile accounts that misuse the service to hide or disguise the origin of funds and to evade monetary record keeping and AML report requirements implemented by an MMT service provider to control mMoney transfers. The fraud chain consists of a sending fraudster, a receiving fraudster and intermediaries, i.e. money mules or smurfs, involved in an *ML activity*. At that, the *length* of a fraud chain is determined by the number of money mules performing fraudulent transactions.

Definition 2 (Money Laundering activity): A Money Laundering (ML) activity of a fraud chain can consist of one or more *ML operations* occurring at arbitrary time intervals during the observation period.

Definition 3 (Money Laundering operation): Each Money Laundering (ML) operation represents a complete structured money transfer between two fraudsters and consists of several individual *ML transactions* between the fraudsters and the mules belonging to the fraud chain.

If fraudsters want to act smart and use different mules for every ML operation, then, the longer the observation period, the higher the length of the fraud chain is.

We consider an ML scenario as depicted in Fig. 2. Fraudsters Mallory and Oscar are end-users of an MMT service. Mallory needs to transfer a large amount X of mMoney to Oscar and does not want to leave any direct traces between their mobile accounts that can be logged by the service platform or trigger

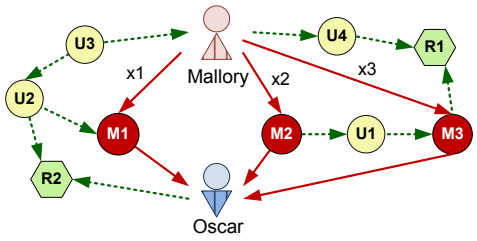


Figure 2. Fraud scenario (smurfing).

an alarm on exceeding transaction amounts. Mallory recruits, e.g. using phishing, n end-users as money mules, who form her network of intermediaries participating in ML. For each ML operation, Mallory selects an arbitrary number of mules from this network and transfers to each mule M_i a small share x_i of the total mMoney amount to be "laundered", such that $\sum x_i = X$. In Fig. 2, the six corresponding ML transactions are denoted with solid (red) lines. Mules keep a small fixed percentage ($\leq 10\%$) of the received mMoney as a service fee and transfer the rest to Oscar.

The explored fraud scenario involves following assumptions: (i) fraudsters and mules can make regular transactions (shown with dotted (green) lines in Fig. 2), such as MD or C2C transfers, along with ML transactions; (ii) mules do not make fraudulent transfers among themselves.

III. State of the Art

A. Fraud Detection Database

There is a lack of publicly available databases for fraud detection [10], [11] that prevents a sound comparison of fraud detection techniques. Most studies on fraud detection [10], [20] are based on private databases, and thus, cannot be used for comparison. This can be easily understood as responsible parties may be reluctant to disclose information about vulnerabilities of their services and are obliged to maintain privacy of their clients. In order to create such databases, several attempts in different fields were undertaken to produce synthetic data [21]–[23]. Simulators from [21], [22] are dedicated to fraud in video-on-demand systems and are not applicable in our case. The synthetic database [23] targets MMT systems as well. However, it does not model the MMT architecture and ML techniques such as smurfing. The simulator introduced in [17], [24] enables the simulation of more complex scenarios such as ML activities. For this reason, we adopt it as a base for the MMT simulator used to create synthetic databases for ML detection (cf. Section V-A).

B. Fraud Detection Techniques

Many techniques have been investigated for fraud detection, mainly from the statistical and data mining field [10]–[13]. The easiest way is to use thresholds on transaction amounts or any other statistical value, such as transaction or expenditure rate [10]. Among the data-mining techniques, mostly used are neural networks, SVMs, Bayesian network models, and naive Bayes scoring. Here, we can also cite decision trees, decision tables and logistic regression which are easier to interpret than neural networks or SVMs. However, well-known industrial actors still use quite old techniques: VISA implements neural networks in the fraud detection tool RST performing real-time scoring of transactions [14]. The M-PESA MMT service has deployed Minotaur™ Fraud Management Solution based on the use of business rules and neural networks [15].

Several machine learning techniques, such as SVM or random forest, have gained prominence in the recent years. Bhattacharyya *et al.* [13] made a comparative study of SVM, random forest and logistic regression on a database of real-life credit card transactions. While SVM and logistic regression show good results, the random forest technique has the overall better performance in their study. Gaber [24] presents a comparative study of several machine learning algorithms on a synthetic database of MMT transactions. The fraud cases relate to theft and malicious infections of mobile phones. The studied algorithms are Bayes net, naive Bayes, SVM, regressions, nearest neighbors, decision table, decision tree and random forest. The best algorithms in this study are PART decision table, C4.5 decision tree and random forest. Lopez-Rojas and Axelsson [23] show that random forest provides better results than naive Bayes classifier and random tree on a synthetic database of MMT transactions containing ML.

The survey by Sudjianto *et al.* [12] covers such methods as SVM, decision trees (CART, C4.5, C5.0), neural networks, Bayesian belief networks, hidden Markov models and link analysis as well as unsupervised techniques, such as anomaly detection and clustering. Bolton and Hand [10] overview fraud detection methods, such as rule-based methods or link analysis. However, these works do not give a clear comparison or recommendations regarding these techniques. A critique of fraud detection techniques concerning their commercial applicability is provided by Phua *et al.* [11]. The two main points are that there is too much emphasis on non-linear supervised machine learning techniques, such as neural networks or SVMs; and that semi-supervised or unsupervised techniques are the only data mining options for future approaches.

To the best of our knowledge, the only application of business process analysis to fraud detection in MMT systems was reported in [25]. A process model reflected transfer habits of end-users. The detection was based on the assumption that for each end-user the transaction amount is limited to a constant range and does not suddenly change. Amount classes were defined and transitions between these classes were monitored. If an abnormal change was observed, an alert was generated and the transaction was labeled as fraudulent. The fraud scenario implied that fraudulent transactions have much lower amounts than the average in the system. The work showed a good performance with the recall of 80%-90% on modeled transactions and 40%-45% on all fraudulent transactions.

IV. Model-based Fraud Detection in MMT Service

A. Design Goals

The state of the art technologies for fraud detection in the banking field may impose substantial limitations when applied to MMT. Since in most cases supervised methods are used, a good training database is vital for the reliable performance. The problem is that for MMT services respective data are usually not available. The chosen fraud scenario poses additional difficulties: fraudsters tend to camouflage ML activities, so that they would be statistically indistinguishable from the behavior of a regular user. We aim to provide an alternative method for fraud detection in MMT services achieving the following:

Recognition performance: The method is intended to support analyst activity and should offer recognition performance comparable to the state of the art. Considering that

parameters of ML operations may be very close to those of legitimate transactions *false positives* are held acceptable and deserving further investigation. *False negatives*, on the contrary, are critical, because fraud committed with an MMT service can have legal implications for its provider.

Usability: The method should increase analyst efficiency. Usability in this case means that the method should be easy to use, give meaningful alerts and reduce total alert volume, offer comparable or better performance than traditional fraud detection techniques. Alert reduction is an important goal because the extensive number of alerts produced by a fraud detection system affects reaction times and hinders its adoption. For the same reason, alerts need to be easily interpretable and signify actionable results. At that, detection delay – the time span between a fraudulent event and its recognition – is a major performance metric and should be minimal.

Autonomy: The method should not depend on availability and relevance of training data and signature bases. For ML, real samples of fraudulent operations are often unavailable. Though simulated data sets can be used instead in some cases (cf. Section III), we believe that such dependency would limit the adoption. Signature-based methods use preset fields that must be met to trigger a rule. As ML schemes are diverse and flexible, the later is also considered to be restrictive.

Next we introduce our method for detection of fraud chains related to ML; its pros and cons in comparison with classical machine learning techniques will be discussed in Section V-D.

B. Predictive Security Analysis at Runtime (PSA@R)

As a basis for the fraud chain detection method proposed in this paper we adopted a model-based approach for event-driven process security analysis PSA@R [16]. Here we summarize the concepts used for the analysis of the MMT system.

The core idea of PSA@R is to validate security compliance of critical processes, evaluating events related to their execution against formally defined workflows and security properties of these processes. It enables identification and management of changes in process behavior as well as early detection of possible security requirement violations for proactive response.

Fig. 3 shows three major phases of PSA@R in the application to an MMT system. Firstly, at the *specification* phase, chosen MMT processes need to be formally defined. At that, three interrelated formal models are created: process model, event model and security model. An operational *process model* is specified using Asynchronous Product Automata (APA), a family of elementary automata [26]. PSA@R uses an operational formal model of a process to compute its expected behavior depending on the observed system state. The process behavior is represented as a directed Reachability Graph (RG) of an APA, whose nodes refer to states and labeled edges to state transitions of the APA. State transitions are driven by (internal) events extracted from the input event stream. An *event* implies that a certain process *action* has been executed resulting in a new state. An *event model* maps real events (MMT transactions) to internal events filtering out information not relevant for security analysis. A *security model* specifies

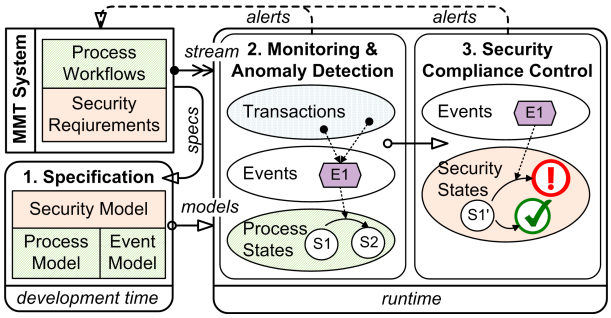


Figure 3. Phases of Predictive Security Analysis at Runtime in MMT system.

process security properties by means of finite-state automata, so-called monitor automata [16], which define a set of security states for each state of the RG. Security critical states of a monitor automaton signify violations of security requirements.

At runtime, PSA@R performs *monitoring and anomaly detection* in the MMT system. In order to verify the actual process behavior, events from the input stream representing actions of the MMT system are checked against the process model of the originating process instance. PSA@R identifies deviations from the expected workflow and produces alerts. For *security compliance control*, PSA@R validates if the actual process behavior meets the specified security properties. If an event triggers a state transition in one of monitor automata representing security properties, the state of the automaton changes accordingly. In case a critical state is reached, a security alert is generated. If PSA@R finds within the prediction scope a possible state transition of a monitor automaton which leads to a critical state it generates a predictive alert (warning).

PSA@R requires as input only descriptions of process workflows that need to be verified at runtime and corresponding security requirements, and does not use any other prior information, such as samples of ML transactions. At that, the underlying method of formal specification allows easy interpretable alerts to be produced during runtime, since they are linked to the states of the process or security model.

For the purpose of ML detection, we extend PSA@R with an additional method allowing to reveal fraud chains. We describe our contribution in the next section.

C. Revealing Fraud Chains

Detection of fraud chains in MMT services is a new application scenario for PSA@R. One of the main challenges in adoption of a model-based approach is to find a proper abstraction level to define processes regarding ML activity, otherwise the performance and usability are affected [25]. For this reason, we had to extend PSA@R, namely, to add a capability to define and synchronize processes on different abstraction levels using synthetic events derived from the monitoring data. Following PSA@R, we describe underlying formal models and then introduce an algorithm which allows us to identify fraudsters among end-users and reconstruct chains of money mules used for ML. Further we refer to the proposed detection method as the FCD (Fraud Chain Detection).

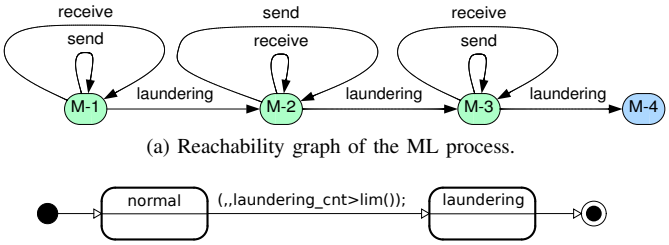
Event Model: We focus on the suspicious behavior of end-users observed from incoming and outgoing transactions on their mobile accounts. Events are propagated when an end-user commits a transaction. The transactions log defines the format of events received from the MMT system and contains for every transaction T the sender s, receiver r and amount a, along with other fields p_i, omitted in the current study. We define the event mapping as follows: $T(s, r, a, p_i) \rightarrow \mathcal{E}(s, r, a)$. Every internal event $\mathcal{E}(s, r, a)$ generates two actions: the $send(\mathcal{E})$ action related to the sender of the transaction T and the corresponding $receive(\mathcal{E})$ action for the receiver of this transaction.

Process Model: We define an abstract ML process that represents ML activity of a fraud chain. Each state of this process refers to a money transfer between two fraudsters made through the mediation of a mule. Fig. 4a shows an RG for the ML process: the more nodes the graph has, the more intermediary-enabled transfers were performed. Thus, the number of the nodes determines the length of a fraud chain. State transitions in this process model are driven by events representing transactions committed in the MMT system. The edges of the RG are labeled with the respective actions $send$, $receive$, and $laundering$ (cf. Fig. 4a). The actions $send$ and $receive$ correspond to an *individual* process that represents the behavior of end-users conducting C2C transfers. Instances of this process are characterized by the user identifier available as an attribute of the event, i.e. s or r. At that, an observed MMT event changes the current process state for both the sender and the receiver. Monitoring of individual processes allows us to single out mule candidates and potential fraudsters, as senders and receivers of transfers made with mWallets of the supposed mules. A *network* process represents the behavior shown by a group of end-users, in our case, by a pair of fraudsters, who organized a fraud chain. An identifier of a network process instance is an identifier of the fraudster pair. Multiple instances of the network process refer to the same state of the abstract ML process. When a new mule candidate appears, the synthetic $laundering$ action is generated, and the respective network process proceeds to a new ML state (see algorithm FCD).

Security Model: We set a security goal for the individual processes as follows: *Mobile accounts in an MMT service owned by end-users must not be used to conduct money transfers on behalf of a third party.* To determine if the observed behavior of an end-user can be rated as ML activity and single out mule candidates we use the following criterion:

Criterion 1: If for an mWallet of an end-user an outgoing C2C transaction $send$ with the amount a_{sent} and a previously committed incoming C2C transaction $receive$ with the amount $a_{rec} : a_{rec} - a_{sent} \leq \Delta_a$ exists, then the user owning this mWallet is labeled as a mule candidate; the sender s of $receive$ and the receiver r of $send$ are labeled as fraudster candidates. The parameter Δ_a is a service fee charged by mules. It can vary between fraud chains, but is constant for all ML operations performed by the same chain. In accordance with [27], we limit the fee to $0 < \Delta_a \leq 10\%$ in our experiments (cf. Section V-B).

For the network processes, we formulate the following security goal: *End-users of an MMT service must not conduct structured money transfers involving intermediate mobile accounts.* This goal is intended to rule out the behavior that helps to disguise the original source and total amount of mMoney transfer. To decide if end-users suspected to be money mules are engaged into the same fraud chain we evaluate the criterion:



(a) Reachability graph of the ML process.

(b) Monitor automaton determining the length of ML fraud chains.

Figure 4. Process model and security model for ML detection

Criterion 2: $f1$ and $f2$ are fraudster candidates. If for each of the two mule candidates $m1$ and $m2$ an outgoing C2C transaction $send : r = f2$, and a previously committed incoming C2C transaction $receive : s = f1$ exists, and $\Delta_a^{m1} = \Delta_a^{m2}$, then both mule candidates $m1$ and $m2$ belong to the fraud chain $(f1, f2)$.

Generally speaking, it is possible that in an MMT service legitimate chains emerge. For example, parents can transfer money to their child, so that s/he is able to pay the rent to the landlord. But the length of such chains will be usually short (in this scenario it equals 1). For this reason, we introduce a *detection threshold* to enable the control over generated alerts. The detection threshold defines the number of intermediaries involved into transfers between two end-users. The higher this number, the more likely the observed activity is ML. The monitor automaton representing this condition is given by Fig. 4b, where $lim()$ refers to the detection threshold.

Algorithm FCD: An RG that describes the common behavior of MMT end-users as well as the behavior of the parties involved into ML is computed (see Fig. 4a). During simulation, multiple process instances reflecting the end-user behaviors and ML activities are assigned to this graph. The detection algorithm verifies whether transitions in this processes occur. As defined in the event model, every transaction (MMT event) is mapped to the internal event $\mathcal{E}(s, r, a)$ that generates the $send(\mathcal{E})$ action and the corresponding $receive(\mathcal{E})$ action. For every $receive(\mathcal{E})$ action the respective transactions are stored in the transactions table RT under the identifier of the receiver r of the event $\mathcal{E}$:

$\mathcal{RT}[r] := \mathcal{RT}[r] \cup \mathcal{E}$

Based on the data stored in the element $\mathcal{RT}[s]$ of the table $\mathcal{RT}$ for every $send(\mathcal{E})$ action ML candidates are determined by means of the function $getlc$. The result of this function is a list of sending fraudster candidates. In the function $getlc$ a heuristics for the candidate selection is implemented based on Criterion 1. In the laundering table $\mathcal{LT}$ all transactions related to the particular fraudster pair $(f1, f2)$ are stored. The function $check\_laundering$ implements a heuristics defining whether the laundering process for a fraudster pair continues, as given

by Criterion 2. The helper functions $getr$ and $gets$ deliver the identifier of the receiver and the sender for a transaction respectively. The pseudo code 1 describes the processing of the $send(\mathcal{E})$ action on an abstract level.

Algorithm 1 (FCD):

for $f1$ in $getlc(\mathcal{E}, \mathcal{RT}[gets(\mathcal{E})])$ **do**
  $f2 := getr(\mathcal{E})$
  $lid := (f1, f2)$
  if $check\_laundering(LT[lid], \mathcal{E})$ **then**
    $LT[lid] := LT[lid] \cup \mathcal{E}$
    $generate\_action(laundering, lid, \mathcal{E})$
  end if
end for
$generate\_action(receive(\mathcal{E}), getr(\mathcal{E}), \mathcal{E})$

The function $generate\_action$ generates a new action (1st parameter) for a given process (2nd parameter) and is responsible for state transitions in the RG for this process.

V. Experiments

A. MMT Simulations

We use the MMT simulator from [17], [24] to generate a database needed to conduct experiments on ML detection. As the simulator presented in [23], this one is based on a multi-agent platform. However, it simulates both the MMT system and users of this system, as well as the habits of end-users.

The simulated platform is made of (1) a front office which interacts with users and processes operation requests and connections to the service, (2) an account management system which controls accounts and processes financial operations, (3) a logs server and (4) a data warehouse which registers the history of the front office and the account management. The payment sequence expects the following pattern [28]: (1) authentication, (2) transmission of sender's payment instructions and transaction details to the MMT platform, (3) authorization by the MMT platform, (4) credit and debit on the receiver's and sender's accounts. A log entry is created when a simulated user carries out a transaction and registered in the transaction database, after the account management system. Each entry contains the transaction type, the transaction amount, the sender and receiver pre- and post-transaction balance, the sender and receiver category, and the transaction date. The generated database contains all simulated transactions for several months.

Three categories of legitimate actors are involved in the MMT system: *End-users*, *Merchants* and *Retailers*. Each category consists of several roles that are associated with specific actions in the platform. *End-users* are individuals who use their mobile devices to access the MMT platform and carry out transactions. *Merchants* sell services or goods to end-users. *Retailers* are in charge of the distribution of electronic money.

The simulation is based on the assumption that legitimate users' transactions are mostly related to their habits. A habit is a repetition of a sequence of legitimate transactions which are characterized by (1) a type of transaction, (2) a normally distributed transaction amount, (3) a normally distributed period of time between two transactions of the considered habit, (4) an initial date and (5) a final date. A user's behavior is composed of a set of habits $H = \{H_1, ..., H_i, ..., H_n\}$, where H_i is a habit for one specific type of transaction. Habits assigned to end-users come from a list of five available habits: Money Deposit (MD), Money Withdrawal (MW), Merchant Payment (MP), Client-to-Client transfer (C2C) and Airtime Recharge (AR). Fig. 5 presents two different configurations for end-users. Their sets of habits differ. They can have common habits that may be configured with different parameters.

(a) End-user 1 (b) End-user 2

Figure 5. Two end-users of MMT system with different habits.

The malicious behavior of fraudsters and mules is also modeled as habits. Thus, fraudsters and mules are selected randomly from the end-users, and the respective habit is added to their habits. Each sending fraudster is associated with a list of mules representing the mules recruited by the fraudster. The behavior of the sending fraudster is to launch ML operations on a regular time basis. To launch an ML operation, she chooses several mules from the list, splits the amount of money to be laundered, and sends the money to the chosen mules within a short interval of time. On receiving the money, a mule transfers it to the receiving fraudster within a day keeping a fee.

Configuration of the MMT simulator: We created 10,000 end-users, who have between 1 and 4 habits. Table I presents the percentage of end-users with respect to the number of habits they have. The table also shows, which habits are associated with end-users depending on the number of their habits. For example, the majority 63.17% of created end-users have only 1 habit and 26.30% have 2 habits. The AR habit is shown by 60.35% of the users with 2 habits. According to these figures, end-users with 1 habit mostly conduct AR and few MD, while those with two or more habits use much more MD and C2C.

From among these end-users, we created 10 fraud chains made of a sending fraudster, a receiving fraudster, and several mules. All these parties are chosen randomly among the end-users. Each fraud chain has a different number of mules, and a varying number of mules is used for ML operations. This configuration is presented in Table II. For example, fraud chain 7 has 7 mules, but only 4 of them, randomly chosen, are used for each ML operation. The fee rate is fixed randomly, but is the same for all mules from the same fraud chain. The sending fraudster conducts an ML operation approximately each month.

Table I
Partition of end-users and habits by the number of habits

	1 habit	2 habits	3 habits	4 habits
Part of end-users	63.17%	26.30%	8.67%	1.86%
MD	11.54%	82.37%	97.66%	98.91%
MW	2.76%	18.86%	46.73%	97.83%
MP	0.22%	2.46%	3.50%	6.52%
C2C	2.79%	35.95%	63.55%	100%
AR	82.69%	60.35%	88.55%	96.74%

Table II
PARTITION OF MULES NUMBER AMONG FRAUD CHAINS

Fraud chain	1	2	3	4	5	6	7	8	9	10
No. mules recruited	3	3	5	5	5	7	7	7	7	7
No. mules used	3	3	3	4	5	3	4	5	6	7

Table III
CONFUSION MATRIX

	Predicted normal	Predicted fraudulent
Actual normal	True negative (TN)	False positive(FP)
Actual fraudulent	False negative (FN)	True positive (TP)

Two databases were created. The first, database A, is used for the training phase of machine learning algorithms. It covers four months of data and contains 272,038 transactions, among which 329 are fraudulent representing 38 ML operations. The second, database B, used for the validation of ML detection techniques, represents seven months and contains 466,359 transactions, with 611 fraudulent ones for 72 ML operations.

Fig. 7a shows when each fraud chain performed an ML operation. Each cross on the figure represents an ML transaction and each cluster of crosses represents an ML operation. We can see that there is approximately a month between the ML operations. The ML activity of each fraud chain lasts throughout the seven months of the simulation.

B. Experiment Setup

The objective of the experiment is to compare the efficiency of several machine learning algorithms and the proposed FCD method for ML detection based on PSA@R, using the same database generated by the simulator. To evaluate the applicability of our model-based method for fraud chain detection we implemented it as an AML plug-in to the process modeling and verification tool PSA [18], [25]. For the machine learning algorithms, we used the Weka toolbox [29]. As presented in section III, we selected the PART decision table [30], the C4.5 decision tree [31] and the random forest algorithm [32].

The machine learning algorithms used a data format aggregated from the original transaction logs. We added fields computed over time to have more information on each transaction. We computed the minimum, maximum, mean, and the total amount of transactions emitted by the sender within a week, as well as the number of transactions, and the number of transactions with the receiver. We did the same data aggregation over a day and an hour. We also kept some original fields, such as the type and amount of the transaction, the sender's and receiver's category, and the date of the transaction. This format was proved to be more efficient for machine learning algorithms than the original one [24].

The machine learning algorithms are trained on database A and tested on database B. The PSA is tested on database B. The performance metrics for the evaluation are the precision and the recall regarding fraudulent transactions. These metrics are extracted from the confusion matrix presented in table III. The precision, computed as $\frac{TP}{TP+FP}$, should be possibly high in order to avoid false alarms that would require time to investigate and might block legitimate end-users in real MMT systems. The recall, computed as $\frac{TP}{TP+FN}$, should also be high, as it means that a fraud chain is discovered faster.

Table IV
ML DETECTION RESULTS FOR MACHINE LEARNING ALGORITHMS

	PART		C4.5		Random Forest	
	N	F	N	F	N	F
Actual normal	465,721	27	465,741	7	465,740	8
Actual fraud	397	214	381	230	385	226
Precision	88.79%		97.04%		96.58%	
Recall	35.02%		37.64%		36.98%	

C. Results

We present results of the three selected machine learning algorithms in Table IV. The figures reveal that the C4.5 decision tree and the random forest work better than the PART decision table. The C4.5 decision tree and the random forest show roughly the same results, with the precision around 97% and the recall around 37%. However, the C4.5 decision tree has slightly better performance. Even though the precision is good, the recall is quite low for all three algorithms. A closer look at this result suggests that for all ML operations at least one transaction is labeled as fraudulent. Fig. 7b depicts the result of the PART decision table algorithm. Each cross corresponds to a true positive, and circle to a false positive. False negatives are not presented for readability reasons.

Comparison of Fig. 7a and Fig. 7b shows that all 72 ML operations have been detected, but not all transactions from an ML operation were labeled. In general, the first two transactions from the sending fraudster to mules are not classified as fraud, neither the transactions from mules to the receiving fraudster. This explains the recall of 35%. Thus, it would require additional efforts to detect the complete fraud chains. Results for C4.5 detection tree algorithm are presented in Fig. 7c. Similarly, each ML operation can be detected with some investigation. Results of the random forest,not pictured here, are very close to those of the C4.5 algorithm.

Table V shows results for the FCD acquired with the detection threshold of 3. The precision and recall are, respectively, of 99.8% and 90.1%, which is a way better than in case of machine learning algorithms. In Fig. 7d, presenting the result for PSA@R, the crosses denote positively labeled transactions. Thus, the cross on the FP line is a false positive, while the others are true positives. The circle corresponds to the negative detection, so the circles on each fraud chain line are false negatives. We can see that with the detection threshold of 3

Table V
ML DETECTION RESULTS FOR THE FCD

	Normal	Fraudulent
Actual normal	465,747	1
Actual fraudulent	60	551
Precision	99.81%	
Recall	90.18%	

the FCD can miss ML operations (chains 1, 2, 3 and 6) if the fraud chain length is lower than this threshold (< 3).

Notice that once a fraud chain is detected, all subsequent transactions are correctly detected as fraudulent. In Fig. 7d a time interval before fraud chain is detected and fraudulent transactions can be identified is denoted as t_{psa}. This interval depends on the chosen detection threshold and can be shortened by lowering its value. We suppose that in this case there might be a trade-off between detection delay and false positive rates, but we could not prove this consideration on our testing data.

We also have investigated the effect of using a sliding time window on the FCD detection capabilities. As the number of end-users in an MMT service is usually high, we addressed a scenario, when the proposed method meets its performance limits. One of possible solutions in such situation is to restrict the number of monitored ML processes to a sliding window, and discard candidates that are outside this window (see Section IV-C). We have made tests with sliding windows of sizes from one to six months (see Fig. 6). In all cases, the FCD was able to correctly detect fraud chains. At that, the smaller the size of the sliding window, the more fraud chains are detected. The reason is that for the same pair of fraudsters all mules involved in transactions over a longer observation period are ascribed to the same chain, while for a smaller sliding window a new chain is created. Thus, the number of detected chains is larger, while the chains themselves are shorter.

D. Discussion

We have shown that both machine learning algorithms and the proposed FCD method enable fraud chains detection related to ML activity. The precision is quite good in both cases, and only few false positives appear, even though simulated ML operations had parameters very close to regular transfers. However, we see several advantages of the new chain detection method based on PSA@R. Firstly, according to the overview of existing techniques for fraud detection presented in Section III, for operational reasons, semi-supervised or unsupervised approaches are to be preferred. This holds in case of PSA@R, in contrast to supervised machine learning algorithms which require a training database to work. In this sense, the proposed method is autonomous, depending only on specifications of processes, whose security compliance needs to be verified.

The FCD also demonstrates better precision and recall than for machine learning algorithms. At that, machine learning algorithms can only label individual transactions leaving the task of fraud chain detection to an analyst, while the FCD singles out fraud chains immediately. This leads to much less effort during the investigation of alarms raised by a fraud detection tool. Such system would then show better usability and be more economic for a bank or an MMT service provider.

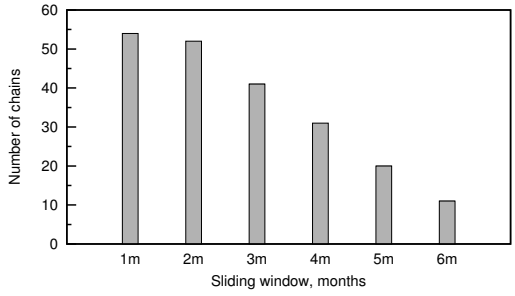


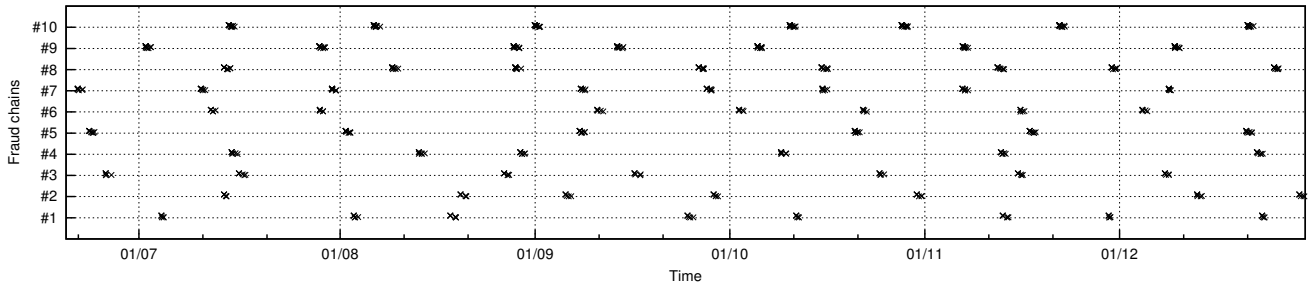
Figure 6. Effect of sliding time window on the FCD fraud chain detection.

We see a potential advantage of machine learning algorithms in that suspicious activity can be detected earlier, although with additional investigation overhead. The comparison of ML detection results for both approaches presented in Fig. 7 shows that in case of the FCD there is a delay between the beginning of ML activity and its recognition. During this time span the respective fraud detection tool would not give any "hints" about the conducted fraud. This delay could be minimized by selecting a proper detection threshold. In this respect, we analyzed the influence of the fraud chain length. We have found that when there are few mules in the chain, it might be more complicated to detect the chain, and lower detection thresholds should be set. However, the higher the amount of money a fraudster wants to transfer, the higher the chain length will be. Thus, a critical ML activity can be detected already with the first ML operation.

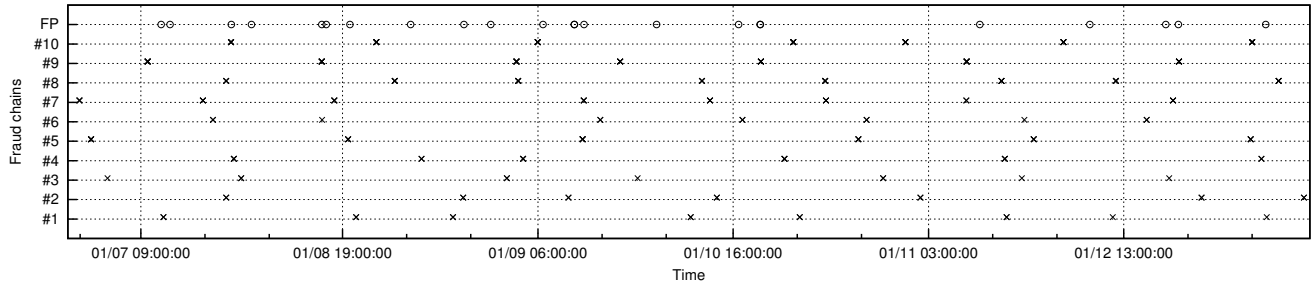
The computational performance of both approaches is satisfying, with analysis of all 466,359 transactions (10,000 end-users) of database B done within minutes on a standard computer. Nevertheless, considering that modern MMT services such as M-PESA number millions of users, we have experimented with time sliding windows as a means to solve performance issues that might occur when the FCD is deployed in a real MMT environment. The experiment proved that using a sliding window in order to reduce the number of simultaneously monitored processes could provide a plausible solution, since it does not affect the recognition performance.

As no public data on ML is available, we evaluated both machine learning algorithms and the FCD method on simulated transaction logs. At that, the use of a simulated database might introduce a detection bias. Indeed, even though we reproduce the normal behavior correctly according to a real-world database, the fraudulent behavior is defined based on the modeled fraud scenario. Such a bias could help to detect the fraudulent cases. However, as all methods used in the comparative study can exploit this bias, the comparison between algorithms is reasonable. An experiment with actual real-world data might present different results, but the order of algorithms regarding their performance should be the same.

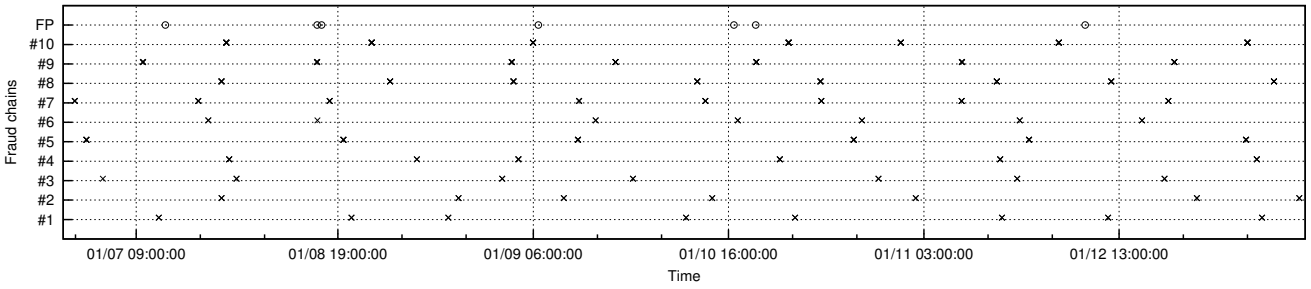
We designed the FCD method bearing in mind the MMT use case. What makes ML techniques such as smurfing possible in MMT systems is the availability and ease-of-use of the MD, C2C and MW operations. These operations are not always available, as with traditional e-banking, in particular, the C2C. However, in case such operations are available, for example, in bitcoin transactions, the proposed detection method could be also applied provided it is tuned for the use case. Indeed, as users of a different service may behave differently compared to the MMT service subscribers, the transactions database might present different characteristics.



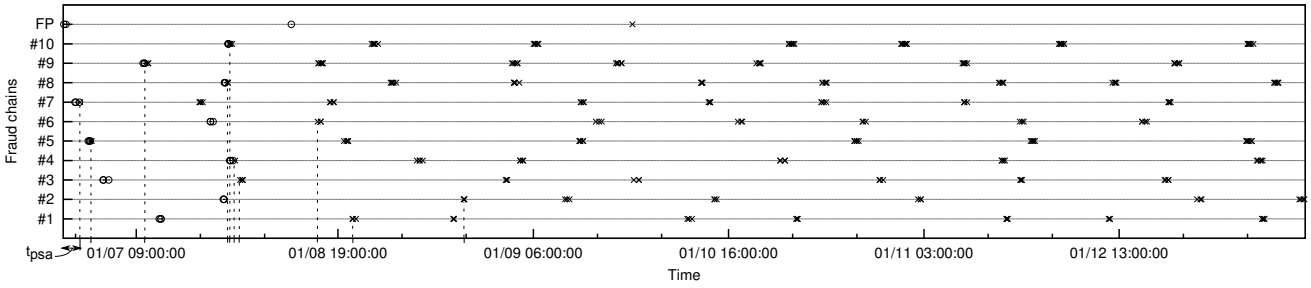
(a) Ground truth: simulated ML transactions.



(b) Detection of ML transactions with PART decision table [30]: crosses denote true positives and circles – false positives.



(c) Detection of ML transactions with C4.5 decision tree [31]: crosses denote true positives and circles – false positives.



(d) Fraud chain detection with the FCD method: crosses denote positively labeled transactions, while circles – negatively labeled; t_{psa} – detection delay.

Figure 7. Comparison of ML detection results obtained with different detection approaches.

VI. Conclusion

The rapid expansion of MMT services is an incentive for fraudsters, and the detection of ML activities is compulsory for MNOs providing these services. Therefore, detection tools are quite important: they must be efficient in terms of detection and easy to use. Semi-supervised or unsupervised detection tools seem to be the way to go, as they do not need a training database that might be difficult to obtain. Thus, classical supervised machine learning tools, even though they demonstrate quite good detection rates, in some application scenarios might not be as efficient as other techniques.

In this paper we proposed an alternative, model-based ML detection method that is able not only to identify individual fraudulent transactions, but detect complete fraud chains, i.e., end-users of an MMT service acting as fraudsters and money mules (or smurfs). This method extends an approach for event-driven process security analysis PSA@R [16] and enables its application to detection of fraud chains in ML scenarios. In order to prove the efficiency of our method we compared it with several classical machine learning algorithms using a synthetic database produced with an MMT system simulator. The recognition performance shown by the FCD method is

better compared both to machine learning algorithms and business process analysis from [25], with precision and recall of 99.8% and 90.1% correspondingly.

Though with the fraud chain detection deployed an MNO would be able to identify and potentially block phone numbers used by fraudsters, the prosecution would be possible only if the owner of this number can be found, which is often not the case [7]. For the future work, we plan to further extend this approach to detect other types of fraud conducted in MMT services, for example, agent frauds, such as split deposits and split withdrawals [5]. We also look into questions related to the integration of the proposed fraud chain detection method with an account management system or an AML system. This includes definition of interfaces and elaboration of models and alert formats, so that the target system could "understand" the received alerts and provide adequate response. Beside that we investigate possibilities to verify our ML detection results in a field study with an MMT service provider.

ACKNOWLEDGMENT

The presented work was developed in context of the project MASSIF (ID 257475) co-funded by the European Commission and ACCEPT (ID 01BY1206D) funded by the German Federal Ministry of Education and Research.

REFERENCES

[1] S. Shen, "Forecast: Mobile Payment, Worldwide, 2013 Update. G00248364," http://www.gartner.com/doc/2484915, Gartner, Tech. Rep., 2013, last visit on 18/06/2014.

[2] C. Pénicaud, "State of the industry: Results from the 2012 global mobile money adoption survey," http://www.gsma.com/mobilefordevelopment/state-of-the-industry-2012, GSMA, Tech. Rep., 2013, last visit on 18/06/2014.

[3] K. Banks, "The invisible bank: How kenya has beaten the world in mobile money," http://newswatch.nationalgeographic.com/2012/07/04/the-invisible-bank-how-kenya-has-beaten-the-world-in-mobile-money/, July 2012, last visit on 18/06/2014.

[4] Orange, "Orange money," http://www.orange.com/en/press/press-releases/press-releases-2012/Orange-Money-reaches-4-million-customers-and-launches-in-Jordan-and-Mauritius, June 2012, last visit on 18/06/2014.

[5] J. L. Mudiri, "Fraud in Mobile Financial Services," http://www.microsave.net/resource/fraud_in_mobile_financial_services, MicroSave, Tech. Rep., 2012, last visit on 18/06/2014.

[6] National Drug Intelligence Center, "Money Laundering in Digital Currencies. No.2008-R0709-003," http://www.justice.gov/archive/ndic/pubs28/28675/, U.S. Department of Justice, Tech. Rep., 2008, last visit on 18/06/2014.

[7] P. Chatain, A. Zerzan, W. Noor, N. Dannaoui, and L. de Koker, *Protecting Mobile Money against Financial Crimes: Global Policy Challenges and Solutions*, ser. Directions in Development. World Bank Publications, 2011. [Online]. Available: http://books.google.de/books?id=jvi9BwJr6TkC

[8] Australian Transaction Reports and Analysis Centre (AUSTRAC), "Money Laundering in Australia 2011," http://www.austrac.gov.au/money_laundering_in_australia_2011.html, Tech. Rep., 2011, last visit on 18/06/2014.

[9] D. Birk, S. Gajek, F. Grobert, and A. Sadeghi, "Phishing phishers - observing and tracing organized cybercrime," in *Internet Monitoring and Protection, 2007. ICIMP 2007. Second International Conference on*, 2007, pp. 3–3.

[10] R. Bolton and D. Hand, "Statistical fraud detection: A review," *Statistical Science*, pp. 235–249, 2002.

[11] C. Phua, V. Lee, K. Smith-Miles, and R. Gayler, "A comprehensive survey of data mining-based fraud detection research," *Artificial Intelligence Review*, 2005.

[12] A. Sudjianto, S. Nair, M. Yuan, A. Zhang, D. Kern, and F. Cela-Díaz, "Statistical methods for fighting financial crimes," *Technometrics*, vol. 52, no. 1, 2010.

[13] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, "Data mining for credit card fraud: A comparative study," *Decision Support Systems*, vol. 50, 2011.

[14] VISA, "Security and trust at every level," http://www.visaeurope.com/en/about_us/security.aspx, last visit on 22/03/2013.

[15] Neural technologies, "Minotaur^{TM} Fraud Detection Software - Finance Sector," http://www.neuralt.com/fraud_detection_software.html, last visited on 18/06/2014.

[16] R. Rieke, J. Repp, M. Zhdanova, and J. Eichler, "Monitoring security compliance of critical processes," in *Parallel, Distributed and Network-Based Processing (PDP), 2014 22th Euromicro International Conference on*, 2014.

[17] C. Gaber, B. Hemery, M. Achemlal, M. Pasquet, and P. Urien, "Synthetic logs generator for fraud detection in mobile transfer services," in *International Conference on Collaboration Technologies and Systems*, 2013.

[18] P. Verissimo *et al.*, "MASSIF Architecture Document," http://www.massif-project.eu/sites/default/files/deliverables/MASSIF_Architecturedocument_v15_final.zip, April 2012.

[19] W. Jack, S. Tavneet, and R. Townsend, "Monetary theory and electronic money: Reflections on the kenyan experience," *Economic Quarterly*, no. 96, First Quarter 2010 2010.

[20] Y. Sahin and E. Duman, "Detecting credit card fraud by decision trees and support vector machines," in *International MultiConference of Engineers and Computer Scientists*, 2011.

[21] E. Lundin, H. Kvarnström, and E. Jonsson, "A synthetic fraud data generation methodology," in *Information and Communications Security*, ser. Lecture Notes in Computer Science, R. Deng, F. Bao, J. Zhou, and S. Qing, Eds. Springer Berlin Heidelberg, 2002, vol. 2513, pp. 265–277.

[22] E. Barse, H. Kvarnström, and E. Jonsson, "Synthesizing test data for fraud detection systems," in *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, 2003, pp. 384 – 394.

[23] E. Lopez-Rojas and S. Axelsson, "Multi agent based simulation (mabs) of financial transactions for anti money laundering (aml)," in *Nordic Conference on Secure IT Systems*. Blekinge Institute of Technology, 2012.

[24] C. Gaber, "Sécurisation d'un système de transactions sur terminaux mobiles," Ph.D. dissertation, Université de Caen Basse-Normandie, October 2013.

[25] R. Rieke, M. Zhdanova, J. Repp, R. Giot, and C. Gaber, "Fraud detection in mobile payment utilizing process behavior analysis," in *International Conference on Availability, Reliability and Security, ARES 2013*. IEEE Computer Society, 2013, pp. 662–669.

[26] P. Ochsenschläger, J. Repp, R. Rieke, and U. Nitsche, "The sh-verification tool – abstraction-based verification of co-operating systems," *Formal Aspects of Computing, The International Journal of Formal Method*, vol. 10, pp. 381–404, 1998. [Online]. Available: http://sit.sit.fraunhofer.de/smv/publications/download/FormAsp.ps

[27] M. Aston, S. McCombie, B. Reardon, and P. Watters, "A preliminary profiling of internet money mules: An australian perspective," in *Ubiquitous, Autonomic and Trusted Computing, 2009. UIC-ATC'09. Symposia and Workshops on*, July 2009, pp. 482–487.

[28] M. Llanes, E. Prieto, R. Diaz, , L. Coppolino, A. Sergio, R. Cristaldi, M. Achemlal, S. Gharout, C. Gaber, A. Hutchison, and K. Dennie, "Scenario requirements (public version)," FP7-257475 MASSIF European project, Tech. Rep., April 2011.

[29] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[30] E. Frank and I. H. Witten, "Generating accurate rule sets without global optimization," 1998.

[31] J. R. Quinlan, *C4. 5: programs for machine learning*. Morgan kaufmann, 1993, vol. 1.

[32] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.