MultiMAuS: A Multi-Modal Authentication Simulator for Fraud Detection Research

Luisa M Zintgraf [(a)], Edgar A Lopez-Rojas [(b)], Diederik M Roijers [(c)], Ann Nowé [(d)]

[(a),(c),(d)]Vrije Universiteit Brussel
[(b)]Norwegian University of Science and Technology

[(a)]lmzintgraf@gmail.com, [(b)]edgar.lopez@ntnu.no, [(c)]droijers@ai.vub.ac.be, [(d)]ann.nowe@como.vub.ac.be

ABSTRACT
MultiMAuS is an agent-based simulator for payment transactions, intended for the analysis and development of dynamic on-line fraud detection methods via a multi-modal user authentication system. The multi-modal authentication procedure allows for a flexible number of authentication steps a user has to do before a transaction is processed (or rejected). It can thus adapt to the risk associated with a certain transaction, in the context of a given user. Our simulator is based on real-world credit card transaction data, to realistically model customer behaviour. The simulator can be used to study short and long term consequences of fraud detection algorithms, for different scenarios like varying levels of fraud or authentication steps. The implementation was done in Python, and is publicly available together with aggregated real transaction data (which serves as input to the simulator) and an example simulated transaction log.

Keywords: multi-modal authentication, fraud detection, multi-agent simulation, credit card transactions

Implementation: github.com/lmzintgraf/MultiMAuS

1. INTRODUCTION

Fraud detection research is committed to the development of methods to detect and prevent fraud. However, researchers are frequently hindered by the fact that there is little publicly available data, and it is often difficult to get access to datasets, due to security and privacy concerns. This means that research in this field is either restricted to the few publicly available datasets, synthetic (potentially unrealistic) data, or private datasets which cannot be shared, making it hard to reproduce results or compare different approaches.
One way to circumvent these problems is to use realistic simulations, based on real data that does not have to be shared together with the simulator (Lopez-Rojas and Axelsson 2012a). Such simulators are able to produce transaction logs which are similar enough to real-world data to allow reliable fraud detection research, and can safely be shared publicly. Besides increased reproducibility and comparability of published fraud detection methods, such simulators have additional benefits over real-world data: first, they allow to generate datasets of arbitrary size, which still follow the distribution of the real (potentially smaller) dataset. Second, it allows to test the reliability of methods when the environment changes, by changing the parameters of the simulator. E.g., we can test what happens when the amount of fraud increases, or when fraudulent behaviour changes over time.
In this paper, we are particularly interested in (developing a simulator for) *on-line* fraud detection, specifically in the context of payment transactions. A central issue with fraud detection methods that are heuristics or trained off-line on fixed datasets is a lack of flexibility when deployed in practice. I.e., the authentication strategies are often *uniform* and *static*: the same rules apply to all transactions, and the method cannot adapt when customer behaviour (of genuine or fraudulent customers) changes over time. There exists a trade-off between customer satisfaction and fraud detection however. Genuine customers prefer convenient methods when possible, especially for low-risk, inexpensive or regular transactions. If too much authentication is asked from them frequently, they might get annoyed and divert to other payment options. We thus need models that can learn how genuine users react to authentication mechanisms in the long term, and adapt the authentication level given the current transaction and user, including her past behaviour. Regarding fraudulent customers on the other hand, static authentication rules give fraudsters the chance to adapt their behaviour to potentially circumvent detection methods (e.g., by exploiting that transactions below a certain threshold will not be checked).
Authentication methods should therefore be *dynamic,* i.e., adapt to changes in user or fraudster behaviour, and at the same time consider the long-term consequences of the authentication procedures. The goal of a fraud

detection method is thus to prevent fraud, but also to keep genuine users satisfied, so that they are likely to make future purchases via the same payment processing platform. Testing such dynamic on-line methods can be challenging in practice, because there is a high risk involved in real-life applications, and it takes time to observe long-term consequences (which might be unforeseeable), and we can usually only test one approach at a time.
We therefore present MultiMAuS, a simulator that offers a tool for on-line fraud detection research, by simulating credit card transactions with multi-modal authentication. The authentication process is dynamic in the sense that the number of authentication steps can vary for different transactions and users. The authentication algorithm can decide how many steps of authentication to request to verify a user's identity. E.g., for online credit card transactions the default authentication is just entering the security code of the card, which can easily be compromised. A second authentication step could be in form of a sms-verification, or calling a customer directly to verify their identity. We use an agent-based modelling framework for the simluator, in which customers, fraudsters and merchants interact by making transactions via a *payment processing platform*. The construction of the simulator consists of two steps: the core simulator is generated based on aggregated data from real online credit card transactions, and represents a system with only the default authentication step (presented in section 4.1). We then extend this to multi-modal authentication, and additionally model the patience and satisfaction of the users (section 4.2).
In the following section, we outline related work to simulators and fraud detection. In section 3, we introduce the properties of the real data we base our simulator on, and what exactly we aggregated from this to use as direct input to the simulator. Section 4 then describes in detail how the implementation was done. The performance of the simulators is analysed in section 5. We compare the real and simulated transaction logs to show that the simulator produces realistic data. Furthermore, we show the performance of a few simple authentication mechanisms for multi-modal authentication in terms of revenue.

2. RELATED WORK

Lopez-Rojas and Axelsson (2012a) highlight in their work the challenges of obtaining data in fraud detection research, and propose using synthetic datasets generated by simulation. They propose multi-agent based simulation, which allows to implement simple behaviour for the participating agents (like customers and fraudsters), while leading to a complex dynamic when these agents interact, exhibiting overall behaviour similar to real-world financial transactions.
Previous work on simulators for fraud detection relying on real data exists for, e.g., mobile money payments (Gaber et al. 2013), shoe retail stores (Lopez-Rojas et al. 2013), and money laundering (Rieke et al. 2013). We refer to Lopez-Rojas and Axelsson (2016) for a more extensive overview of the use of simulations in fraud detection research regarding financial datasets. Amongst others, Haq et al. (2016) present a novel technique to automatically generate synthetic data. This however only produces data for batch learning. For an on-line setting on the other hand, we explicitly need independent agents (customers and fraudsters) which make transactions sequentially, and where we can intervene with fraud detection methods.
To the best of our knowledge, our simulator is the first to support multi-modal authentication, and customers changing their purchase behaviour based on their experience. We think it is therefore uniquely suitable for developing and assessing dynamic, flexible on-line fraud detection algorithms.
Our simulator is intended for fraud detection research, and can be used to test both off-line and on-line methods. Surveys of off-line methods are, e.g., Kirkos et al. (2007) or Wang (2010). Research specifically into dynamic, adaptive methods is, amongst others, Alese et al. (2012) and Wheeler and Aitken (2000).

3. DATA

Our simulator is based on real credit card transaction data, provided by a company that processes online credit card payments (and whose identity we cannot disclose). The data consist of around 90,000 non-fraudulent (genuine) transactions and around 1,200 fraudulent transactions from 2016. Each transaction has the following attributes: credit card ID, merchant ID, amount, currency of purchase, date of transaction, card issuing country, fraud label. The labels were obtained by fraud reports made by the credit card companies.

3.1. Data Summary

The number of unique credit cards in the dataset is around 54,000 for genuine, and 800 for fraudulent transactions. Each card is used on average 1.65 times, and most credit cards are only used once (around 65%). Customer activity depends heavily on the hour of the day (most transactions occur in the evening), and the month of the year (most transactions occur in summer). On average, 10.18 non-fraudulent and 0.13 fraudulent transactions are made per hour, which is the time step we chose for our simulator. The transaction amounts range from about 0.5 to 7,800 Euro (after converting everything to the same currency). Purchases are made with credit cards from 126 countries (19 for fraudulent transactions) in 5 (3) different currencies. There are a total of 7 merchants (after removing merchants with less than 100 transactions), of which 6 are affected by fraud. A complete overview of the data properties is given in the results section 5 (table 1), where we compare the real data to simulated data.

3.2. Aggregated Data

To realistically model customer and fraudster behaviour, we aggregated information from the real dataset. Our simulator is built on these, and the aggregated data can be (unlike the raw transaction logs) shared publicly without compromising the privacy of the real customers. In this section we summarise what kind of data is used in our simulator.

Customers. The following information was extracted for genuine and fraudulent customers (separately).

- The total number of transactions in the entire dataset, i.e., in the year 2016.
- The (empirical, discrete) distribution of credit cards over countries (i.e., what is the probability of a customer/fraudster having a credit card from a certain country).
- The distribution over currencies, given the country.
- The distribution of transactions over the merchants, conditioned on the currency. This is to determine where a customer will purchase something.
- The expected number of transactions, given: the hour of the day, the day of the week, the day in a month, and the month in a year.
- The probability of a customer/fraudster making another transaction, given she just made one.
- The average number of days between two subsequent transactions (which is 12.99 days for genuine and 8.94 for fraudulent customers).

To estimate the probability of a customer making another transaction (after just having made one), we only looked at those that did a transaction in the months April and May of 2016, and then estimated how many of these did at least another transaction until the end of 2016. We did this irrespective of how often this card was used again to simplify the computations. The reason we only look at April and May is that the cards used during the first few months of 2016 were *never* used again. We believe this is due to some peculiarity of the data (there were not enough customers, or data is simply missing), since the repetition pattern starting around April is very different, and many genuine customers make several transactions with their cards. For the cards used later in 2016 we do not have enough future data to estimate whether more transactions were made. Due to these restrictions to April and May, we will have more average transactions per card than in the original dataset. This is intended and we think is realistic behaviour of customers. It is important for us to model the transaction patterns of unique customers over time, to simulate how they adapt their behaviour to different authentication models.
Additionally to the above, we estimated the fraction of compromised cards that were used both in fraudulent and genuine transactions in this dataset (around 33%). This will later be used to determine whether fraudsters steal credit card details from existing users, or users outside of the customer pool. Further, we estimated the probability of a genuine customer making another transaction after the card was compromised (which lies only at about 9%).

Merchants. For the simulator, we use the merchants from the true dataset with more than 100 transaction. All merchants with less than 100 transactions were not subject to fraud, and they do not play a noticeable role compared to the other merchants. We do not know whether this is because of a lack of data for these merchants, or whether there is an intrinsic property that makes these merchants less subjective to fraud. One of the remaining seven merchants was also not subject to fraud. For each of these merchants we model the distribution over the price range of their products/services (separately for genuine and fraudulent customers) by using histograms (with 20 bins).

This aggregated information is used as input for the simulator, and released together with our implementation. In the following section, we explain how our simulator utilises the information to simulate transaction behaviour of genuine/fraudulent customers and merchants.

4. MODEL AND IMPLEMENTATION

The simulator was implemented in Python 3 and uses the agent-based modelling framework mesa (Masad and Kazil, 2015). Our implementation is available at github.com/lmzintgraf/MultiMAuS.
In this section, we describe in detail how the simulator was implemented. We do so in two steps; we first present the base implementation with a uni-modal authentication model (section 4.1., UniMAuS), and then the extension of this to incorporate multi-modal authentication (section 4.2., MultiMAuS). We separate the description because the simulator with uni-modal authentication is based on the real transaction logs, and we can then directly analyse whether the simulated data has similar patterns (which we do in section 5.1.).

4.1. UniMAuS

In this section, we describe our base implementation UniMAuS (Uni-Modal Authentication Simulator) - i.e., when there is only one (implicit) authentication step, as in the real transaction logs. By default, all incoming

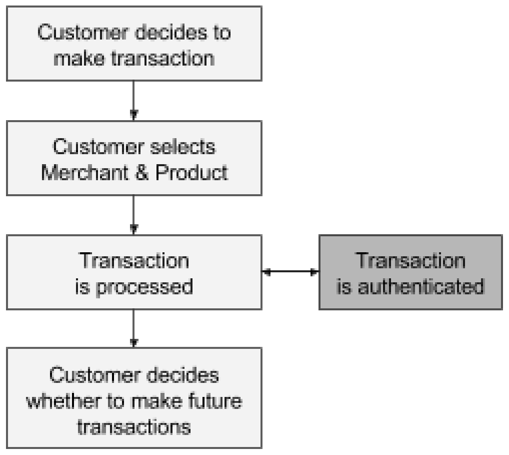
transactions are permitted and processed without further authentication. This part of the simulator then produces transaction logs that follow patterns similar to the original data. It can be used to produce data sets with realistic properties, of arbitrary size, over any time span. Additionally, different scenarios can be simulated, such as varying levels of fraud.
This base implementation can be used for on-line learning where transactions can either be permitted or denied (without further authentication), or for off-line learning with the produced transaction logs. In our context, its main intention is to serve as a realistic basis for incorporating multi-modal authentication (section 4.2.).

4.1.1. Entities

There are two active agents in the simulator, genuine customers and fraudulent customers (and later also the authenticator, see section 4.2., MultiMAuS). At each time step, these (autonomously) decide whether to make a transaction. Merchants are passive agents, who offer their products to customers, but do not actively execute any actions during the simulation. In the initialisation phase, merchants and customers are created, based on the real transaction data. We describe the details of these agents in the following.

Merchants. We set the number of merchants to the number of merchants observed in the real dataset, restricted to ones with more than 100 transactions. Each merchant further has a sampling function to generate prices. Whenever a customer/fraudster picks a merchant to purchase a product from, the amount for this product is sampled according to this distribution (which differs for genuine and fraudulent customers). Sampling takes place by first randomly selecting a bin for a price range according to the probabilities taken from the histograms of the real data. We then uniformly at random select an amount within this selected price range.

Customers (fraudulent/genuine). Each customer gets a unique credit card ID when instantiated. Further we assign a country to the customer, sampled according to the empirical (discrete) distribution over countries. Conditioned on this country, we select a currency (again, sampled from the empirical distribution). Finally, each customer has a scalar value we call the ‘intrinsic transaction motivation’. This is for all customers set to 1 divided by the total number of customers in the simulation. This way we can make sure that the total number of transactions is approximately the same as in the true data, and have customers that do decisions *autonomously* (without looking at what other agents do, or having a central mechanism that decides which customers make transactions).



Customer decides to make transaction
Customer selects Merchant & Product
Transaction is processed
Transaction is authenticated
Customer decides whether to make future transactions

Figure 1: Action flow during one simulated hour. The step “Transaction is authenticated” is optional in the MultiMAuS model. In the UniMAuS model, each transaction is processed without further authentication.

Fraudulent customers. When credit card IDs for fraudsters are initialised, the card ID is sometimes (according to the fraction observed in the real data) taken from the pool of genuine customers. In that case, the country and currency are taken from that customer (the country must be the same since it the card issuing country, and the currency only changed in 3 out of 250 cases in the real dataset, so we chose to disable that option). Otherwise, country and currency are assigned like described above.

The intrinsic transaction motivation makes sure that the total number of transactions is (with the default parameters and little noise) similar to what we observed in the real data. By varying the starting number of customers in our simulator, we can influence how much time lies between two transactions of one customer (the more customers exist, the less often an individual customer will have to make a transaction so that we match the true total number of transactions). We found empirically that starting with 3333 genuine and 44 fraudulent customers works well in this respect. Note that the number of customers can change over time, but that the intrinsic transaction motivation (which in this case is 1/3333 and 1/44) stays the same. This means that when more customers use the payment processing platform, we also get more transactions.

4.1.2. Transactions

Transactions are made sequentially, where one time step represents one hour in Pacific Standard Time. Figure 1 shows a simplified overview of the steps during one such hour. In the following, we describe this in detail.

1. *Start transaction.* Each customer (genuine or fraudulent) decides whether to make a transaction or not. This is conditioned on the current *local* time of the customer (i.e., we

convert the current global time according to the customer's country) and the probabilities of making transactions from the true data (section 3.2). This is then weighted by the intrinsic transaction motivation. The resulting transaction probability is used to (stochastically) determine whether a user will make a transaction.

2. *Pick product.* Each customer/fraudster that decides to make a transaction then picks a merchant. This is done by sampling a merchant from the empirical distribution, conditioned on the currency. In a next step, the amount of the product is determined by sampling from the amount distribution of the merchant.
3. *Process transaction*. After all customers decided whether to make a transaction, these are processed in random order and log entries for each transaction are made. In UniMAuS, no fraud detection mechanism or extra authentication is executed or intervenes, i.e., all transactions will be processed. This also means that all attempted fraudulent transactions will succeed.
4. *Customer migration*. When all transactions have been processed, customers decide whether they will make a transaction in the future (then they stay in the customer pool) or not (in which case they are removed from the customer pool) - again, based on the empirical probability of making another transaction. For genuine users, staying also depends on whether their credit card has been used in a fraudulent transaction or not. There are also new customers added to the customer pool. To compute how many new users join, we use the expected number of users leaving, which is determined by the transaction probability given the *global* time, and the intrinsic motivation of customers/fraudsters. The size of the user pool therefore varies over time, but will stay around the initial numbers 3333 (genuine customers) and 44 (fraudulent customers) with the default parameters and little noise.

4.2. MultiMAuS

Our simulator is intended for the development of *dynamic fraud detection* systems. To this end, we extend our base implementation to support multi-modal authentication. In this section we explain the additional components that were used to realise this.

4.2.1. Entities

Authenticator. The authenticator handles the security measurements taken for each transaction. I.e., the authenticator can evaluate each transaction, and then decide to permit it, request additional authentication, or deny the transaction. For our current implementation, we give the authenticator the option to either permit the authentication, or ask for an additional authentication which is either provided by the user or denied (in which case the transaction gets cancelled). Whenever the second authentication is provided, the authenticator permits the transaction. Note that adaptations to this could be that the authenticator has the option to deny transactions, or that authentications have some sort of quality measure (e.g., how close is a signature to the original signature of the user?). We plan to implement and studies such scenarios in future work.

Genuine customers. Genuine customers are extended to support multi-modal authentication in the following way. During initialisation, a customer gets a *patience* and a *satisfaction* value (both values between 0 and 1). The patience is initialised randomly from a right-skewed beta distribution. The patience gives an indication of how likely the user is to accept more authentications, which will also depend on the transaction amount (see next section).
The satisfaction at the beginning of the simulation is set optimistically for all customers (for our example experiment, we used 0.9). For customers that are added to the pool of customers later in the simulation, it is set to the mean satisfaction of all customers in the pool to mimic the influence of existing customers on new customers via sharing opinions. The satisfaction of the user changes over time with the experiences the user makes with the service. The satisfaction then has an effect on how likely a user is to make additional transactions in the future. How exactly the patience and satisfaction influence the transactions is explained in the next section, 4.2.2.

Fraudulent customers. The properties of fraudster do not change compared to the uni-modal scenario, except that they can now also react to additional authentication (see next section). In the current implementation, fraudsters will always deny a second authentication, since we assume they cannot provide it.

4.2.2. Transactions

In this section, we describe what changes regarding the transactions described for the uni-modal authentication (section 4.1.2.).

1. *Start transaction.* What changes compared to the uni-modal authentication scenario is that the satisfaction of a genuine customer now also plays a role when deciding whether to make a transaction or not. In practise, the transition probability which is computed for the UniMAuS simulator is multiplied by the customer's satisfaction.

1. *Pick Product.* (Same as above.)
2. *Process Transaction.* For each transaction that a customer wants to make, the authenticator can now decide whether to request an additional transaction. Genuine customer will provide this with a probability that is computed by taking the average of the customer's patience and the current transaction amount divided by the maximum possible amount at the given merchant. Fraudulent customers will always cancel the transaction if asked for a second authentication.
3. *Agent migration.* The probability of making another transaction is now multiplied by the satisfaction for genuine customers. I.e., satisfied customers will be more likely to make additional transactions in the future.

The customer's satisfaction changes after every transaction, depending on their experience. If the transaction was successful and there was no additional authentication, the satisfaction goes up by 1%. If however the user provided a second authentication, the satisfaction goes down by 1%. If the user cancels a transaction instead of providing a second authentication, the customer's satisfaction goes down by 5%. We chose these numbers so that there is a clear challenge in making good authenticators that balance the customer satisfaction and pure fraud detection. We show the behaviour of simple authenticators in section 5.2. Unfortunately real data on customer behaviour regarding the reaction of second authentications (cancelling or providing the authentication) was not available to us during the development of this simulator, and we hope to get access to and incorporate real data regarding this in the future.

5. RESULTS

In this section we analyse the performance of our simulator; first to see how well the UniMAuS implementation fits the real data, and second how the MultiMAuS implementation reacts to different simple authentication mechanisms.

5.1. Evaluation: UniMAuS

We compare the real transaction logs (which are private and cannot be shared) to simulated transaction logs from one run with default parameters. These are published together with the implementation, see files 0_transaction_logs.csv and 0_parameters.pkl.
Table 1 shows an overview of the properties from the transaction logs, of the real and the simulated outputs. Running a simulation of one year takes around twenty minutes. We can see that the simulated transaction logs have properties that are close to the real dataset. The largest difference is in the maximum transactions for

Table 1: Overview of summarised data statistics from the real versus simulated transaction logs, taken from a single run of the simulation for the year 2016.

	Customers		Fraudsters	
	real	simul.	real	simul.
transactions	89,194	88,603	1,163	1,239
trans./hour	10.15	10.09	0.13	0.14
trans./month	7432.8	7383.6	96.9	103.3
num cards	54,133	44,817	799	826
num cards, single use	34,873	22,548	543	553
num cards, multi use	19,260	22,269	256	273
fraud cards in genuine	-	-	33%	34%
min amount (€)	0.44	0.44	1	1
max amount (€)	7,835.6	6,276.3	2,737.0	2,685.1
avg amount (€)	298.5	305.6	61.7	69.2
num merchants	7	7	6	6
countries	126	105	19	15
currencies	5	5	3	3
max trans/card	109	17	15	7
avg trans/card	1.64	1.98	1.46	1.50

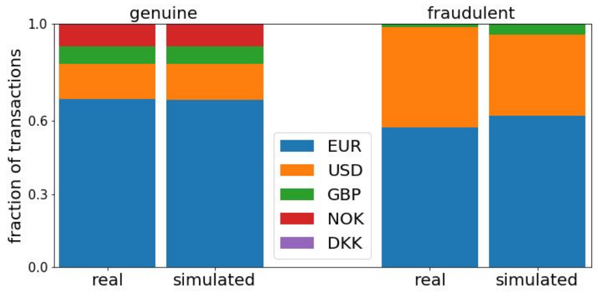


Figure 2: How transactions are distributed over the different currencies. The two columns on the left show this for genuine transactions, comparing the real vs the simulated data. On the right we show this for fraudulent transactions. Note that the currencies NOK and DKK are not present in the fraudulent dataset.

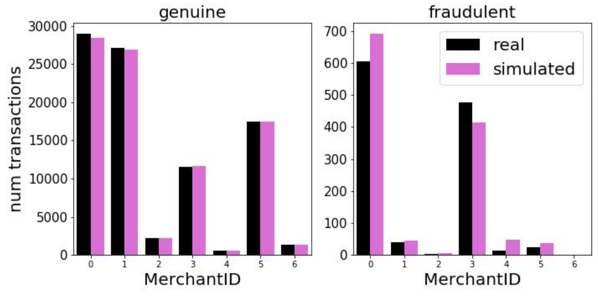


Figure 3: Total number of transactions per merchant in 2016, for genuine (left) / fraudulent (right) transactions. We compare the transaction logs of the real data set (black) and one simulated dataset (magenta).

one single card. Since after each transaction, a customer decides whether to make another transaction (based on true recurrence rates), making as much as 100 transactions per card is unlikely, which is in part due to the way we computed the recurrence rates (i.e., not looking at the number of recurrency but only binary indication, see section 3.2).

Figure 2 shows the distribution of all transactions over currencies. There is a clear difference between genuine and fraudulent transactions: the fraction of transactions in US dollars is much higher in fraudulent transactions. Further, fraud is only done in the three most popular currencies. We also see that the real and simulated data have matching behaviour, with the noise evening out more for (the large number of) genuine transactions.

Figure 3 shows the total number of transactions per merchant (for 2016). The transaction varies a lot over the merchants, with some merchants processing only a few hundred transactions in the entire year 2016, and others several tens of thousands. This is true for both the genuine and fraudulent transactions. The figure shows that this pattern remains also in the simulated data.

Figure 4 shows the the distribution over transaction amounts, summarised for the year 2016 (in Euro, after converting everything to this same currency). About half of all genuine transactions are worth under 100 Euro. Interestingly, the amounts of fraudulent transactions are generally lower, with only about 10% of all transactions over 1000 Euro. This could be due to fraud detection mechanisms (and detected fraud not showing up in our data), or because fraudsters anticipate higher security measures for higher payments.

Figure 5 and 6 show the transaction activity over time, for the month in a year (figure 5) and the hour of the day (figure 6). We can see that the true and simulated data show similar transaction behaviour over time on average. Note that the simulator follows the real data distribution on *average*, but that individual customers (green lines) can differ. This is because when initialising customers, we add some noise to the transaction probabilities to simulate individual behaviour. This noise is higher on small time scales (like hours or days in the week) than on large time scales (like months in a year). In practise this is achieved by sampling from a multivariate Gaussian distribution with the mean being the distributions observed in the real transaction logs and a small variance (of 0.1 in the default case).

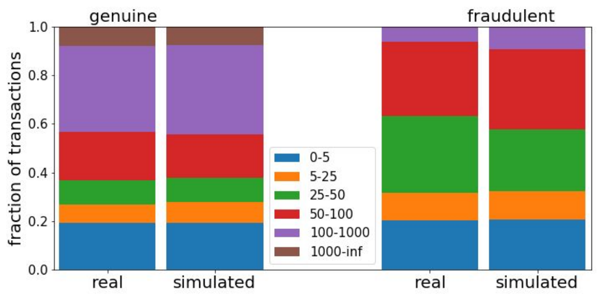


Figure 4: Distribution of transaction amount in Euro). The two bars on the left show the real and simulated data distribution for genuine transactions. The two bars on the right show this for fraudulent transactions.

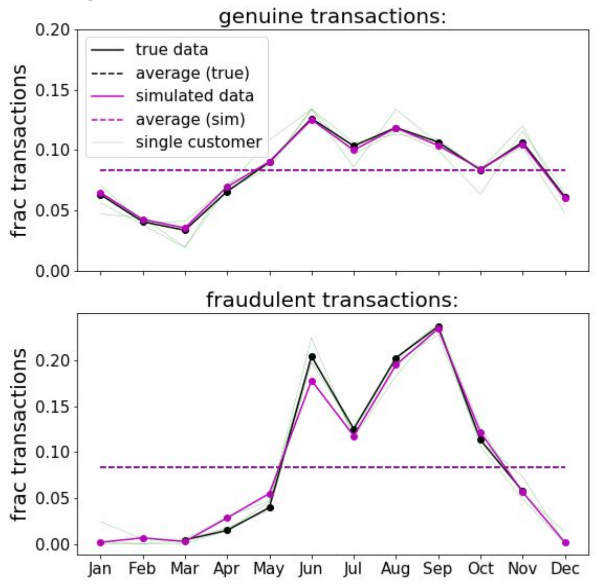


Figure 5: Fraction of transactions per month in a year, for the real data (black) and the simulated data from one run (magenta). We show a few examples of this distribution for a single user in the simulator.

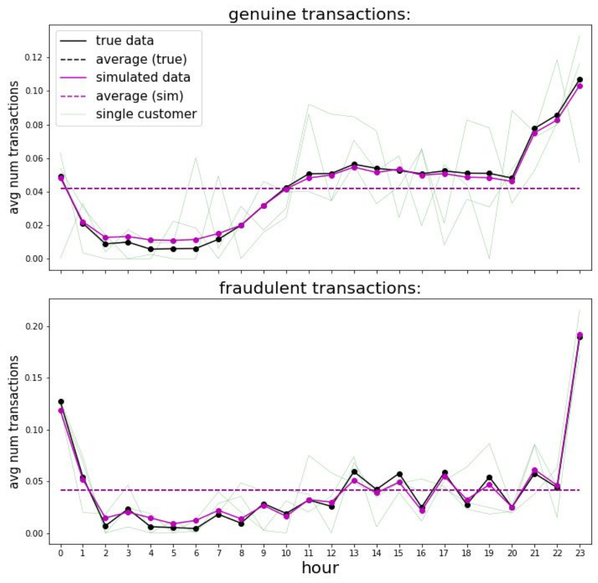


Figure 6: Fraction of transactions per hour in a day, for the real data (black) and the simulated data from one run (magenta).

5.2. Analysis: MultiMAuS

We tested several simple authentication mechanisms on the multi-step authentication simulator. We look at the problem from the view of the payment processing platform, i.e., in terms of business revenue, and calculate the reward of an authentication mechanism as follows. For each successful genuine transaction, the reward is 0.3% of the transaction amount, plus 1 cent (all calculated in Euro). If a fraudulent transaction is not detected, the reward is the negative amount of the transaction, since this has to be paid back to the genuine user whose credit card was compromised.

Figure 7 shows the results in terms of reward from one experimental run, where each run is initialised with the same random seed. We tested five different simple authentication mechanisms, which are the following.

- *Oracle.* The oracle agent has access to the true labels. It never asks a genuine customer for a second authentication, and always asks fraudulent customers for a second authentication. This means that fraudsters will always cancel the transaction (since they cannot provide further authentication), and that user satisfaction is as high as possible. In other words, this is an unrealistically good policy, that provides us with an upper bound on the attainable revenue.
- *Random.* The random authenticator asks for a second authentication with a probability of 0.5. If the second authentication is provided, the transaction is authorised.
- *Heuristic.* The heuristic authenticator only asks for a second authentication if the transaction amount lies above 50 Euro.
- *Never second.* This agent permits all incoming transactions without further authentication. This leads to high satisfaction in the users and therefore relatively more transactions. However as fraud increases (as it does in the summer), the monetary loss outweighs the gain.
- *Always second.* This agent always asks for a second authentication for all transactions. This means that all fraud will be prevented, but that customers also get unsatisfied over time.

As we can see in figure 7, the agent that never asks for a second authentication has the lowest cumulative reward. This is due to the increased number of fraud during summer (where it starts to decline) and because the customer satisfaction does not grow fast enough to make up for this. The authenticator which always asks for a second authentication gains more cumulative reward, but is still significantly below the oracle agent. This is due to the lower customer satisfaction, leading to customers making less transactions.

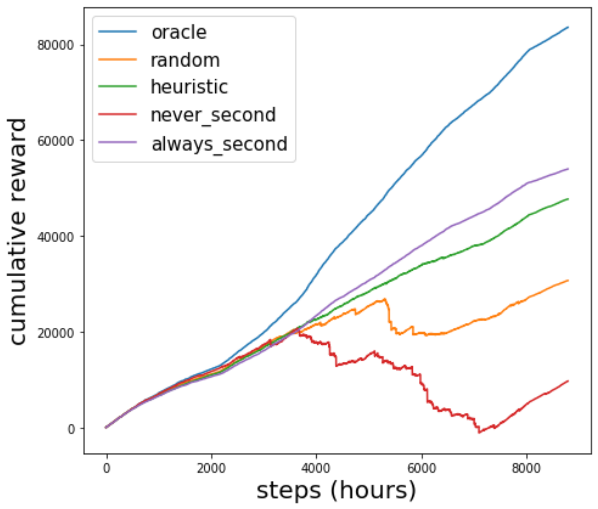


Figure 7: Cumulative reward for different simple authentication agents over one simulation run (one year, where each time step is one hour).

6. CONCLUSION AND FUTURE WORK

We introduced MultiMAuS, a simulator supporting multi-modal authentication systems, which can be used for fraud detection research. We used an agent-based framework, including merchants and customers (genuine and fraudulent) which interact with merchants via a payment processing platform. The simulator is based on real-world data for uni-modal authentication, and on top we added the option for multi-modal authentication. Our results show that the simulated transaction logs produce data which follows similar patterns as the original data. We conclude from this that our implementation realistically simulates real transaction behaviour of genuine and fraudulent customers, and is therefore suited for fraud detection research.

Looking at several simple authentication mechanisms, we see that there is need for more sophisticated (learning) algorithms that fill the gap between the oracle and the other authenticators. We believe that reinforcement learning (RL) algorithms are well suited for this, since they can take into account long-term effects of their actions (like the effects on customer satisfaction and resulting transaction behaviour). There are several interesting directions for future research using the presented simulator that we want to investigate.

One interesting line of future work is in the context of safe RL (Garcia and Fernández 2015). When deploying learning algorithms in practise, we want to make sure the policy acts within reasonable bounds and follows certain constraints regarding security. E.g., we need to have some guarantees on their performance in the sense that we do not want them to perform (much) worse than the policy currently used in practise (e.g., a heuristic policy) which we want to replace (Thomas 2015).

One way of approaching the fraud detection problem with multi-modal authentication is treating it as a multi-objective decision problem (see, e.g., White and Kim 1980, Roijers and Whiteson 2017). I.e., we explicitly try optimising not only the cumulative reward but at the same time the satisfaction of the user. User satisfaction can indirectly be measured by the number of authentication steps required for transactions, or for example by a (sparse) reward signal via direct user feedback (e.g., through user surveys).

We also plan to extend the simulator by implementing adversarial fraudsters that actively adapt their behaviour to circumvent the security measures to commit fraud, leading to a multi-agent setting (Vlassis 2007, Nowé et al 2012, Littman 1994). This is important to study how algorithms can deal with concept drift.

Another aspect to consider is that the customer's satisfaction cannot be observed directly, and thus the problem can be formulated as partially observable (Kaelbling et al. 1998), possibly in combination with multiple objectives (Roijers et al. 2015) and/or multiple agents (Oliehoek and Amato 2016).

We believe there are more interesting opportunities for future work, and hope to provide a useful tool for the development of fraud detection algorithms. We would like to note that since the customer behaviour in terms of satisfaction and patience was not build on real data, we would advise researchers to test algorithms on different initial satisfactions, or update rules for this. We will try to in the future incorporate expert knowledge or information from real data on customer behaviour in this respect.

ACKNOWLEDGMENTS

This research was supported by Innoviris, the Brussels Institute for Research and Innovation, Brussels, Belgium. Diederik M. Roijers is a postdoctoral fellow of the Research Foundation - Flanders (FWO). We would like to give special thanks to the company (identity undisclosed) that gave us access to their data.

REFERENCES

Alese BK, Adewale OS, Aderounmu GA, Ismaila WO, Omidiora EO. 2012. Investigating the Effects of Threshold in Credit Card Fraud Detection System. In International Journal of Engineering and Technology, 2.

Gaber C, Hemery B, Achemlal M, Pasquet M, Urien P. 2013. Synthetic logs generator for fraud detection in mobile transfer services. In Collaboration Technologies and Systems, International Conference, pp. 174-179. IEEE.

Garcıa J, Fernández F. 2015. A comprehensive survey on safe reinforcement learning. Journal of Machine Learning Research, 16 (1), 1437-80.

Haq IU, Gondal I, Vamplew P, Layton R. 2016. Generating Synthetic Datasets for Experimental Validation of Fraud Detection. In Conferences in Research and Practise in Information Technology, 170. Fourteenth Australian Data Mining Conference, Canberra, Australia.

Kaelbling LP, Littman ML, Cassandra AR. 1998. Planning and acting in partially observable stochastic domains. Artificial intelligence, 101(1), 99-134.

Kirkos E, Spathis C, Manolopoulos Y. 2007. Data mining techniques for the detection of fraudulent financial statements. Expert systems with applications, 32(4), 995-1003.

Littman ML. 1994. Markov games as a framework for multi-agent reinforcement learning. In Proceedings of the eleventh international conference on machine learning, 157, 157-163.

Lopez-Rojas EA, Axelsson S. 2012a. Money laundering detection using synthetic data. In The 27th annual workshop of the Swedish Artificial Intelligence Society (SAIS), 071, 33-40. Linköping University Electronic Press.

Lopez-Rojas EA, Gorton D, Axelsson S. 2013. RetSim: A ShoeStore Agent-Based Simulation for Fraud Detection. In 25th European Modeling and Simulation Symposium, 25-34.

Lopez-Rojas EA, Axelsson S. 2016. A review of computer simulation for fraud detection research in financial datasets. In Future Technologies Conference (FTC), 932-935). IEEE.

Masad D, Kazil J. 2015. Mesa: An Agent-Based Modeling Framework. In 14th PYTHON in Science Conference, 53-60.

Nowé A, Vrancx P, De Hauwere YM. 2012. Game theory and multi-agent reinforcement learning. In Reinforcement Learning, 441-470. Springer Berlin Heidelberg.

Oliehoek FA, Amato C. 2016. A concise introduction to decentralized POMDPs. Springer International Publishing.

Rieke R, Zhdanova M, Repp J, Giot R, Gaber C. 2013. Fraud detection in mobile payments utilizing process behavior analysis. In Availability, Reliability and Security (ARES), 2013 Eighth International Conference, pp. 662-669. IEEE.

Roijers DM, Whiteson S, Oliehoek FA. 2015. Point-Based Planning for Multi-Objective POMDPs. In IJCAI 2015, 1666-1672.

Roijers DM, Whiteson S. 2017. Multi-objective decision-making. Synthesis Lectures on Artificial Intelligence and Machine Learning.

Thomas PS. 2015. Safe reinforcement learning. Thesis (PhD). University of Massachusetts Amherst

Vlassis N. 2007. A concise introduction to multiagent systems and distributed artificial intelligence. Synthesis Lectures on Artificial Intelligence and Machine Learning, 1(1), 1-71.

Wang S. 2010. A comprehensive survey of data mining-based accounting-fraud detection research. In Intelligent Computation Technology and Automation (ICICTA), 1, 50-53. IEEE.

Wheeler R, Aitken S. 2000. Multiple algorithms for fraud detection. Knowledge-Based Systems, 13(2), 93-9.

White CC, Kim KW. 1980. Solution procedures for vector criterion Markov decision processes. In Large Scale Systems, 1(4), 129-40.