A Machine Learning Framework for Adaptive FinTech Security Provisioning

Hyun Jung La[1], Soo Dong Kim[2]

[1] Workday, Inc., Pleasonton, USA
[2] School of Software, Soongsil University, Korea
{hjla80, sdkim777}@gmail.com

Abstract

FinTech services bring an elevated level of security concerns due to the non-conventional characteristics such as diverse and evolving transaction models. Hence, conventional financial security provisioning approaches have limited applicability, rather, it requires more effective, intelligent, and reactive anomaly management for FinTech transactions. We present a comprehensive framework for managing FinTech transactions which utilizes machine learning-based intelligence in deriving anomaly detection models and adaptive FinTech security provision. And, we define a formal model of the anomaly management, and present a software framework implementing the model.

Keywords: FinTech, Transaction anomaly, Machine learning, Adaptive security

1 Introduction

Financial Technology, FinTech, refers to new processes, products or business models in the financial services industry, all of which would bring increased values and innovation in financial services [1]. In contrast to conventional financial services, FinTech services reveal the following non-conventional characteristics;

- Diversity in Transaction Models
- Evolvability of Transaction Models
- Customer-centric Transaction Models
- Simplified and speedy Transaction Processing
- Mobile/Wireless Network-based Dataflow

Due to the characteristics, the anomaly management for conventional financial transactions such as credit card payment and mobile payment is not sufficient enough to manage the dynamic and diverse anomaly patterns of FinTech transactions [2]. It requires a new level of effectiveness, intelligence, and timeliness in managing the anomaly of FinTech transactions.

Our approach to FinTech anomaly management is to utilize two technology advances; machine learning in deriving anomaly detection model and adaptive security provision. The adaptiveness is delivered by two features of our framework; timely refinement of FinTech anomaly model and personalization of the anomaly model.

The timely refinement of the anomaly model is essential due to the dynamic nature of FinTech services. And, the personalization of anomaly model is essential since the transaction behaviors of FinTech service customers largely vary.

In this paper, we present a conceptual and software framework for efficiently managing FinTech anomaly. We define an effective and seamless process of adaptive FinTech anomaly provisioning, and we present the formal representation of essential artifacts of the framework. We define key algorithms of the process, which utilize machine learning capability and adaptive nature of methods.

The paper is organized as follows. Section 2 is to survey representative related works, and section 3 is to define the process model of the proposed FinTech anomaly management and formal representation of key artifacts used in the framework. Section 4 is to specify the algorithms used in the process, and section 5 is to present an experiment result with the proof-of-concept implementation of the framework. With the proposed framework, it is highly feasible to manage FinTech anomalies in a timely and adaptive manner.

2 Related Works

Ahmed's work presents survey results of applying clustering algorithms to financial domain [3]. This work defines assumptions on how to detect anomalies and summaries works applying partition-based and hierarchical-based clustering algorithm. Khac's work [4] and Chang's work [5] apply k-means algorithms to detect money laundering behaviors and online auction frauds respectively. Glancy's work [6] and Torgo's work [7] utilize hierarchical clustering to detect anomalies in financial transactions.

*Corresponding Author: Soo Dong Kim; E-mail: sdkim777@gmail.com
DOI: 10.3966/160792642018091905026

West's work presents survey results of applying classification algorithms to financial fraud detection [8]. This work analyzes strengths and limitations of classification-based approach to financial fraud detection, and classifies existing works in terms of performance, applied algorithms, and fraud types. Bhattacharyya's work presents comparison results of applying SVM, random forest, and logistic regressions to a credit card fraud detection [9]. It turns out that the algorithms yield different performances based on the experiment settings. Chang's work proposes a method for early fraud detection in online auctions [10]. They reduce attributes used for generating learned models through principal analysis, and utilize the last 20% of the transaction histories in building the models to maximize detection rates while minimizing efforts.

In addition, there are works of applying hybrid approaches to maximize the fraud detection performance. Behara's work applies fuzzy c-means clustering algorithm and neural network algorithm to detect credit card frauds [11]. Sahin's work utilizes artificial neural network and logistic regressions to detect credit card frauds [12]. Yaram's work utilizes document clustering and classification algorithms for identifying frauds in insurance claims [13].

Existing works focus on selecting an optimal set of features for detecting frauds, identifying frauds from the proposed models, and evaluating performance of the models. The effectiveness of the proposed methods largely depends on the data used for learning models. If learned models come from user-specific data, they may miss typical fraud cases. And, in the opposite case, they tend to less consider personalized anomaly detections. Moreover, existing works need to reflect FinTech-specific characteristics.

3 Computing Model and Artifacts

In this section, we define the process of provisioning FinTech anomaly management, and specify the representations of the key artifacts.

3.1 Process of Adaptive Security Provisioning

We define a process for adaptive security provision as shown in Figure 1.

Activity 1 is to learn a *Global Anomaly Model (GAM)* from the training set of *FinTech Transaction Log*. GAM consists of patterns of FinTech anomaly instances.

Activity 2 is to personalize the GAM with two sources; *Personal Transaction Model (PTM)* and GAM. PTM describes the key characteristics of user's previous FinTech transactions. The resulting *Personalized Anomaly Model (PAM)* is a refined GAM which is adapted to each user.

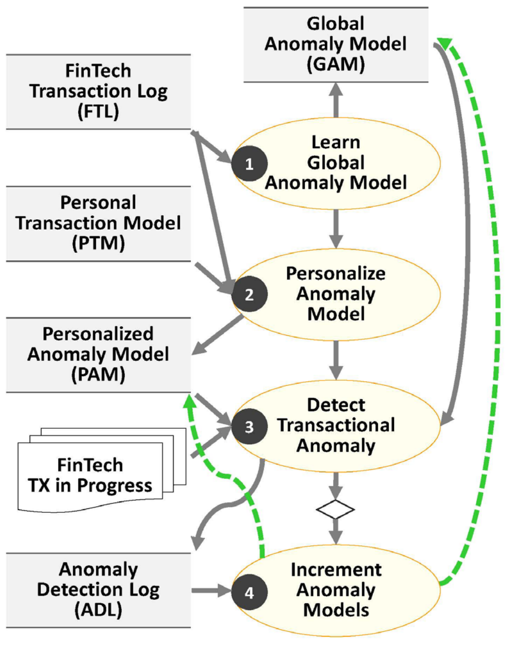

Global Anomaly Model (GAM)
FinTech Transaction Log (FTL)
1
Learn Global Anomaly Model
Personal Transaction Model (PTM)
2
Personalize Anomaly Model
Personalized Anomaly Model (PAM)
3
Detect Transactional Anomaly
FinTech TX in Progress
Anomaly Detection Log (ADL)
4
Increment Anomaly Models

Figure 1. Four activities of adaptive security provisioning

Activity 3 is to detect transactional anomaly from the *FinTech transactions in Progress* using GAM and PAM. In case of any anomaly detection, *Anomaly Detection Log (ADL)* is appended with the result.

Activity 4 is to incrementally learn anomaly model using *ADL* and update GAM and PAM. This activity is to adopt the already learned anomaly models with newly detected transactional anomaly in timely manner. Note that the dashed arrows in Figure 1 indicate the incremental updates on the two anomaly models.

3.2 Adaptiveness in the Process

Adaptiveness of the FinTech security provisioning is provided by two features of our framework; reflection of personal transaction behavior and timely refinement of security anomaly models.

Two activities are performed for providing adaptiveness provisioning. In activity 1, the personal FinTech transactional behaviors are taken into consideration and GAM is elaborated into PAM, which is more specific to each user.

In activity 4, the results of detecting transactional anomaly in activity 3 are taken into consideration and both GAM and PAM are refined with the newly detected anomaly. Hence, the activity 4 is invoked as a polling; invoking the activity only when a certain condition is met.

3.3 Representing "FinTech Transaction Log"

FinTech Transaction Log (FTL) is a repository of FinTech Transactions. Each FinTech transaction is defined as a tuple of *TX ID*, *TX Context*, and *Result* where *TX Context* consists of instances of transaction context elements, and each transaction context is a pair of *tag* and *value* as shown here;

- FTL = (TX_Instance)0..*
  TX_Instance = (TX_ID, TX_Context, Normality, Annotation);
  TX_Context = (Tag, Value)1..*

The *Normality* field indicates the normality of transaction results, and is a value of an enumerated type of (normal, abnormal, abort). The *Annotation* field describes the details of the anomaly in the transaction such as any determined fraud type, cause, and accountability reference. The typical tags for *TX_Context* are shown in Table 1.

Table 1. Typical tags for *TX Context*

Tag	Description	Examples
User ID	Identification Information of Users	554-73-1678 (USA) 91707481(MSISDN)
Account Num	Account Number used in the Transaction	FPA-222-33-68345
TX Type	Type of FinTech Transaction	Loan, Fund Transfer, Request P2P,
TX Amount	Amount of Transaction	$250.00
TX Time	Time of Transaction	2016-11-28T07:55:12Z
TX Env	Environmental context of transaction	Server, IP, Traffic, Public Event, ….

Optional tags include GPS Location, Language Code, and IP Address. An example of *TX_Instance* is given here;

- TX_Instance = (ST_ON_1604_2138879752, {(Account Num, S5547316D), (TX Type, Transfer), (TX Amount, $1,000), (TX Time, 2017-03-28T07:55:12), (IP Address, 203.253.22.100)}, 'Normal', "Unregistered Client Device").

The example consists of four elements as specified in FTL scheme, and *TX Context* has five instances of (tab, value) scheme. The transaction was terminated normally.

3.4 Representing 'Global Anomaly Model'

Global Anomaly Model (GAM) specifies known patterns for identifying anomaly instances. This is generated from the Fintech Transaction Log (FTL) with 'Normality' tags, which can serve as the *labels* for machine learning of activity 1.

Each anomaly pattern in GAM is a tuple of *Anomaly Type*, *Anomaly Context*, *Annotation*, and the probability of anomaly when the Anomaly Context is met.

- GAM = (A_Pattern)$^{0..*}$
  A_Pattern = (P_Type, P_Context, Annotation, Certainty)
  P_Context = (Tag, Value)$^{1..*}$

P_Type is an identification number of each anomaly pattern, *P_Context* is a description of the transaction context for the pattern, *Annotation* is an additional description of the pattern, and *Certainty* is the degree of confidence that the anomaly pattern will occur when the *P_Context* is met. Some of the known *P_Types* are Identity Theft, Embezzlement, Phishing, Fund Transfer Scam, and Counterfeit Identification.

An example of *A_Pattern* is given here;

- A_Pattern = ('Identity Theft', {(Age Group, Youth), (Login Duration, 20+ sec), (Authentication Repetition, 5+), (Region, NK3), (Device, Mobile Phone) }, "A stolen ID was used with unregistered device", 0.75)

In the example, 'Identify Theft' is the anomaly pattern type, the *P_Context* consists of five tag-value pairs, an annotation is given, and the certainly for this pattern is 0.75. That is, the confidence of occurring this anomaly pattern is 0.75 when the given *P_Context* is met.

3.5 Representing "Personal Transaction Model"

In conventional anomaly detection methods, a population-level anomaly models are constructed from a large set of transactions. Hence, the acquired anomaly model is applicable to a large population. However, the drawback of anomaly management with the global anomaly model, i.e. GAM, is that user-specific transactional behavior is not considered enough in deriving GAM. To remedy the drawback, we calibrate GAM with user's specific transaction behavior, i.e. *Personal Transaction Model (PTM)*. The anomaly management with calibrated GAM, i.e. *Personalized Anomaly Model (PAM)*, will result in a more accurate and effective detection of anomaly.

PTM represents each user's FinTech transactional behavior, and it is defined as a set of feature and value pairs.

- PTM = (Feature, Value)

Feature is an attribute which represents user's personal characteristics such as preferred transaction type, average transaction amount, amount range, transaction frequency, regularity of client devices used, and occupation. An example of *PTM instance* is given here;

- PTM Instance = { (Preferred_TX_Type, Balance Transfer), (Ave_TX_Amt, $150), (Amt_Range, $20-$500), (Device Regularity, 85%), (Occupation, Musician) }

In the example, *PTM Instance* consists of 5 pairs of (Feature, Value) description.

3.6 Representing "Personalized Anomaly Model"

Personal Anomaly Model (PAM) has essentially the same representation of GAM. However, PAM has user-specific calibrated values for the attributes of GAM. That is, the anomaly-related context of GAM is tailored by analyzing the PTM, as described in section 3.5

Recall that GAM is a set of anomaly patterns, and each pattern has four elements;

- GAM = (A_Pattern)$^{0..*}$

  A_Pattern = (P_Type, P_Context, Annotation, Certainty)

  P_Context = (Tag, Value)$^{1..*}$

PAM includes the same content of the given GAM, but the tags of *P_Context* could have adjusted values for the tags of PTM.

An example of *PAM instance* is given here;

- PAM Instance = ('Identify Theft', {(TX_type, Payment), (Time Zone, not CDT), (Device, Public Browser), (Amount, \$500+), (TX Freq, 2+/day)}, 0.8);

This example is derived from two sources; GAM and PTM, and it specifies a personalized anomaly instance. If the transaction type is *Payment*, the time zone is not *CDT*, the client device is a publicly available browser (like the one in airports), the transaction amount is over \$500, and the transaction frequency is 2 or more per day, then there is a 0.8 probability of occurring 'Identify Theft'.

Note that PAM instance is specific to a single user, reflecting the personal profile and transactional behavior specified in PTM. From PAM instances, we can infer the typical transaction patterns of a user.

3.7 Representing "Anomaly Detection Log"

Anomaly Detection Log (ADL) is the result of running activity 3, and it is a log of FinTech transactions which are detected as anomaly. ADL is a list of anomaly occurrence cases.

- ADL = (A_Instance)$^{0..*}$

  A_Instance = (User_ID, TX_ID, TX_Type, DateTime, Amount, TX_Context, Anomaly Pattern, Annotation)$^{0..*}$

  TX_Context = (Tag, Value)$^{1..*}$

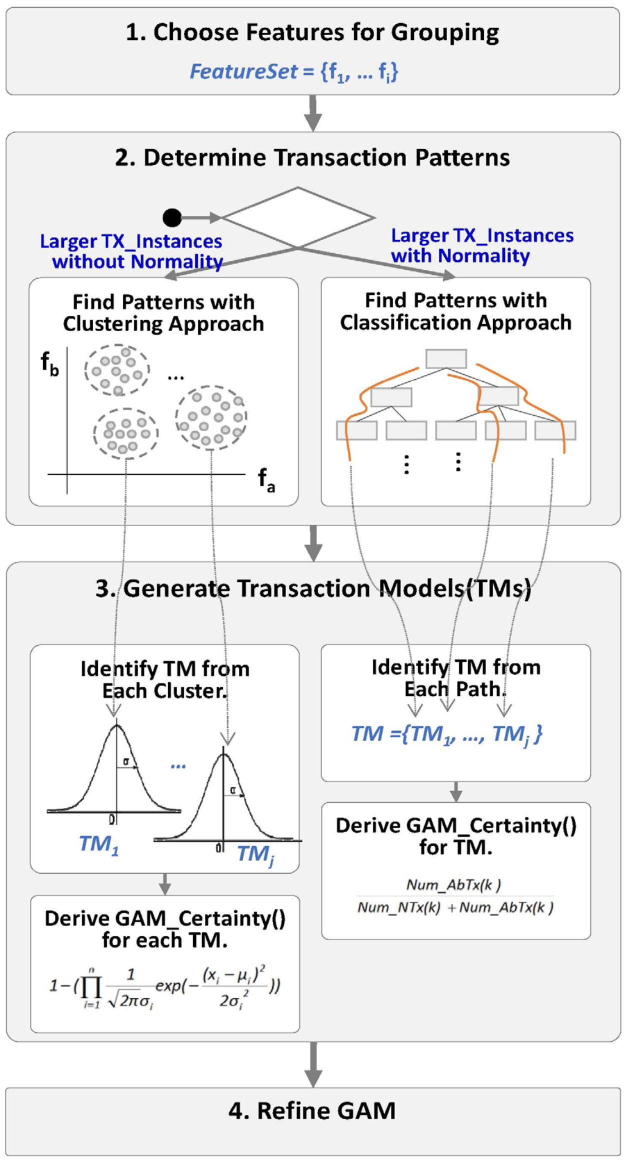
Each anomaly instance *A_Instance*, captures the essential information about a translation detected as anomaly. The ADL is further utilized in incrementally updating GAM and PAM in timely manner, of which algorithm is presented in section 4.4.

4 Design of Algorithms

4.1 Activity 1. Learning Global Anomaly Model

To detect abnormality, it is prerequisite to have knowledge on normal as well as abnormal data [2, 14]. If collected Fintech transactions are largely pre-labelled with the abnormality, it is ideal to apply classification algorithms, to identify similar transaction patterns [15-16]. In practice, normal behaviors are constantly evolved with newer types of transaction mechanisms, and fraudster's tricks are also upgrading. With the classification algorithms, it is limited to find out more types of the transaction patterns automatically. Hence, we also consider clustering algorithms, especially hierarchical clustering algorithm, which can be executed without knowing labels in advance [15-16].

The algorithm for this activity is performed in four steps as shown in Figure 2.



1. Choose Features for Grouping
FeatureSet = {f1, … fi}
2. Determine Transaction Patterns
Larger TX_Instances without Normality
Larger TX_Instances with Normality
Find Patterns with Clustering Approach
fb
fa
Find Patterns with Classification Approach
3. Generate Transaction Models(TMs)
Identify TM from Each Cluster.
TM1
TMj
Identify TM from Each Path.
TM ={TM1, …, TMj }
Derive GAM_Certainty() for each TM.
Derive GAM_Certainty() for TM.
Num_AbTx(k) / Num_NTx(k) + Num_AbTx(k)
4. Refine GAM

Figure 2. Algorithm of learning GAM

Step 1 is to choose features to consider based on the observations on discriminating transaction patterns. In this step, the following features are utilized;

- *Personal profile data* including Age, Gender, Occupation, and Yearly Salary
- *Transaction data* including TX Type, TX Amount, and TX Time

These features are mapped to Tag of *TX_Context*, a field of *TX_Instance*. And, these features are drawn from analysis of consumption trend reports.

Step 2 is to determine transaction patterns which group similar transactions by clustering algorithms and/or classification algorithms. If *FTL* is largely labelled with abnormality-related data, it is better to apply a classification algorithm in this step. Decision tree is considered in this framework. Otherwise, clustering algorithm is appropriate. Since there are different types of transactions in Fintech domain, a portion of labelling with normality in *FTL* is first checked, and an appropriate algorithm is determined.

In case of a hierarch clustering algorithm, each cluster implies similar fintech transactions and consists of abnormal and normal transactions with similar features. And, in case of a classification algorithm like decision tree, each path traversing from a top to a leaf is a transaction pattern, and it contains multiple transactions.

Step 3 is to generate transaction model(s). This transaction model is used to estimate a certainty of being classified as an abnormal transaction.

If a decision tree is applied, a transaction model is generated from each path, and the certainty of a transaction, *x*, is estimated as following;

$$GAM\_Certainty_k(x) = \frac{Num\_AbTx(k)}{Num\_NTx(k) + Num\_AbTx(k)} \quad \textbf{(1)}$$

x is a transaction data which traverses the *k*th path, and *Num_AbTx(k)* and *Num_NTx(k)* returns the number of abnormal transactions and normal transactions belonging to the *k*th path, respectively. Correspondingly, $GAM\_Certainty_k(x)$ returns the portion of the abnormal transactions out of the total transactions in the *k*th path.

If a clustering algorithm is applied, there is no way to know the normality of each data in the cluster. Hence, a transaction model for a cluster is generated as a form of *Gaussian* model, where data deviating from a mean of the cluster has a high possibility of being abnormal. And, the probability density function of the *Gaussian* model is used to estimate the abnormality certainty, as following;

$$GAM\_Certainty_k(x) = 1 - \left(\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{(x_i-\mu_i)^2}{2\sigma_i^2}\right)\right) \quad \textbf{(2)}$$

x is a transaction data and represented with the *n* number of features, such as $x = \{x_i \mid i = 1, 2, ...n\}$, and μ and σ indicate a mean and a standard deviation of the distribution, respectively. Since a higher probability density value indicates that it is close to a mean, the certainty is acquired by subtracting a probability density value from 1.

Step 4 is to refine *GAM* by analyzing transaction patterns and their transaction models. If *GAM* has been already constructed with the domain knowledge, it checks whether *GAM* includes transaction patterns generated in the previous step. If GAM has *A_Pattern* which consists of the same set of features as the transaction patterns, *Certainty* of *A_Pattern* is modified by refining an original value by considering a result of equation (1) or (2). Otherwise, a transaction pattern is added to *GAM* as *A_Pattern*.

In case of applying a classification algorithm, *A_Pattern* is derived as follows;

- *P_Type* is automatically assigned with a sequential number, and later can be edited by human experts.
- *P_Context* is derived from a combination of the feature values in each path. More specifically, a feature used in a decision node and its value are mapped to *Tag* and *Value* in *P_Context*.
- *Annotation* is manually filled by human experts.
- *Certainty* is filled with the result of an equation (1).

In case of applying a clustering algorithm, multiple instances of *A_Pattern* are generated from a transaction model, and *A_Pattern* is derived as follows;

- *P_Context* is derived by aggregating values held by each feature in the model. More specifically, an axis label and its value are mapped to *Tag* and *Value* in *P_Context*.
- *Certainty* is filled with the result of an equation (2).
- *P_Type* and *Annotation* are filled with the same way as the case of applying a classification algorithm.

4.2 Activity 2. Personalize Anomaly Model

The algorithm for leaning *PAM* from *FTL and* PTM is performed in two steps.

Step 1 is to choose features which are effective to discriminate a user's transaction abnormality. In this framework, we consider the following features;

- *Transaction data* reflecting a user's transaction pattern including TX Type, TX Amount, TX Amount, and Merchant Type (e.g. shopping, food, travel, etc)
- *TX Environmental data* (i.e. TX Env) including GPS and Device Type

Some features are derived from Tag in *TX_Context* of *FTL*, and others are from Feature of an element in *PTM*.

Step 2 is to generate *PAM* by building a *Gaussian* model with the features decided in the previous step. The generated *Gaussian* model represents a set of normal distributions for all features selected. Hence, each instance of *PAM* is derived from a combination of abnormality for each of the features. For example, one *PAM* instance explains a situation where features other than 'amount of the transaction' have normal values. And, similar to *GAM*, the *Certainty* field of *A_Pattern* is estimated with the equation (2).

4.3 Activity 3. Detecting Transactional Anomaly

Detecting transactional anomaly is done by comparing *GAM* and *PAM* with transaction data being evaluated. That is, this activity checks whether the current transaction data show an unusual out-of-pattern behavior. The overall algorithm for this activity is depicted in Figure 3.

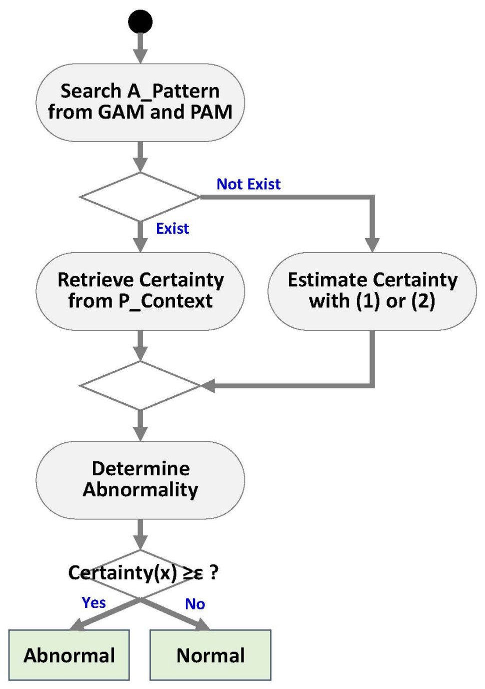

Search A_Pattern from GAM and PAM
Not Exist
Exist
Retrieve Certainty from P_Context
Estimate Certainty with (1) or (2)
Determine Abnormality
Certainty(x) ≥ε ?
Yes
No
Abnormal
Normal

Figure 3. Algorithm of detecting transactional anomaly

First, *A_Patterns* in *GAM* and *PAM* are retrieved with *TX_Context* of the current *TX_Instance*, *x*.

If *A_Patterns* is retrieved from *GAM* and/or *PAM*, whose *P_Context* is same as *TX_Context* of, *x*, it is quite straightforward to acquire anomaly certainty. This is why all *A_Patterns* in *GAM* and *PAM* are described with certainty values. Hence, *GAM_Certainty(x)* and *PAM_Certainty(x)* are values of the *Certainty* field in *A_Pattern* of *GAM* and *PAM*, respectively.

If any *A_Patterns* are not found *GAM* and *PAM*, we need to look for the most similar path (in case that *A_Pattern* is derived from a path in a decision tree) or the closest cluster (in case that *A_Pattern* is derived from a Gaussian model of a cluster). And, *GAM_Certainty(x)* and *PAM_Certainty(x)* are estimated with equation (1) and (2), respectively.

With those certainty values, we determine an abnormality certainty as follows;

$$\begin{aligned} Certainty(x) = {} & GAM\_Certainty(x) \times W_{GAM} \\ & + PAM\_Certainty(x) \times W_{PAM} \end{aligned} \tag{3}$$

Here, W_{GAM} and W_{PAM} are weight values of *GAM* and *PAM* respectively. Rules for defining W_{GAM} and W_{PAM} are listed as follows;

- The sum of W_{GAM} and W_{PAM} is 1.
- If *PAM* is null, W_{PAM} is set to 0.
- Normally, W_{PAM} is larger than W_{GAM} since the abnormal certainty largely depends on user-specific transaction behaviors.
- W_{PAM} and W_{GAM} are adjusted by testing *GAM* and *PAM* with test data, a part of *FTL* and *PTM* which are not used in generating the models.

And, the abnormality of *x* is determined with equation (4).

$$Abnormality(x) = \begin{cases} 1 & \text{if } Certainty(x) \geq \varepsilon \\ 0 & \text{if } Certainty(x) < \varepsilon \end{cases} \tag{4}$$

Here, ε is a threshold value for determining the abnormality and is adjusted with test data, a part of *FTL* and *PTM.*

Finally, *ADL* is generated with the detection results. For example, *TX_Context* and *Anomaly Pattern* are derived from *P_Context* and *A_Pattern* in *GAM* or *PAM*, respectively.

4.4 Activity 4. Increment Anomaly Model

This activity is performed to minimize time to construct *GAM* and *PAM.* The larger transaction data are collected, the more time to construct *GAM* and *PAM* is required. To resolve this issue, the framework refines *GAM* and *PAM* based on the anomaly detection result.

Existence of *A_Pattern* having same features as the current *TX_Instance*, *x*, is checked in the previous activity. Depending on the existence, *GAM* and *PAM* are refined in different ways.

If *GAM* and *PAM* includes *A_Pattern* which is the same as x, the values of *Certainty* of *A_Pattern* in *GAM* and *PAM* are refined by considering the newer *GAM_Certainty(x)* and *GAM_Certainty(x)* respectively.

Otherwise, *GAM* and *PAM* are augmented with a new *A_Pattern* as follows;

- *P_Contexts* in *GAM* and *PAM* are derived from *TX_Context* in *TX_Instance*. Depending on the *Tags* in *TX_Context*, either *P_Context* in *GAM* or one in *PAM* is added.
- Certainty in *GAM* and *PAM* are filled with *GAM_Certainty(x)* and *GAM_Certainty(x)* respectively.
- *P_Type* and *Annotation* are filled with the same way as the Step 4 of the Activity 1.

5 Implementation and Experiments

5.1 Design and PoC Implementation

The framework consists of components and databases as shown in Figure 4.

Anomaly Model Learner learns and derives *GAM* from the transaction set. *Anomaly Model Personalizer* personalizes *GAM* with *PTM* and *GAM.*

Transaction Monitor detects transactional anomaly from the current FinTech transactions using *PAM.* When detecting an anomaly, *ADL* is appended with the newly detected anomaly result.

Anomaly Model Optimizer incrementally learns the anomaly model using *ADL* and updates *GAM* and *PAM.* This activity is to adopt the already learned models with newly detected transactional anomaly.

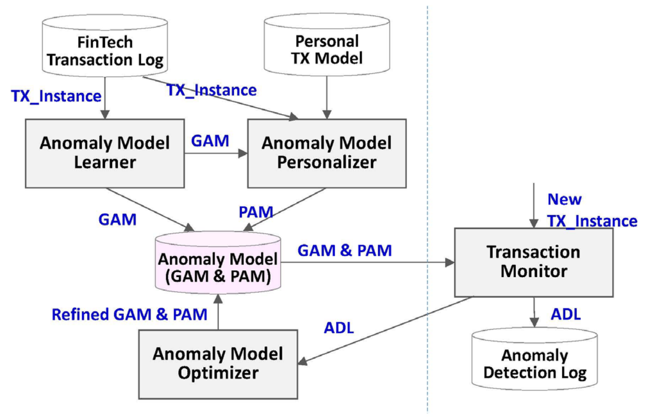


Figure 4. Components of the framework

Our proof-of-concept system was implemented with *Tomcat 8.3* on *Ubuntu 16.04* for the web server, *MySQL* for the databases, and Java. We adopted the web interface for the PoC implementation, and hence we used JSP, JavaScript, and jQuery framework.The key components were implemented based on the algorithms given in section 4. Implementation of the databases is a strict realization of the formal representations given in section 3. Hence, our PoC implementation is consistent with the computing model and the algorithms.

5.2 Experiment Datasets and Results

The real life FinTech datasets are hardly available to public for privacy issues. However, we could acquire a real dataset of 1,250 transactions of 78 users from a finance IT Company providing FinTech mobile payment services in Korea. The identification information was removed from the dataset, and randomly generated ID numbers replaced the identifications. Among 1,250 transactions, there were 18 transactions labeled as 'Abnormal'.

We adapted the dataset based on the representation of *FTL* and conducted 4 experiments for different purposes.

Experiment 1 is to evaluate the effectiveness of the proposed process and algorithms. Using a test driver, we fed the whole dataset into the system, and acquired GAM of 21 anomaly patterns, PAM of 32 anomaly patterns, and *ADL* of 15 detected anomaly instances. The result is shown in Table 2.

Table 2. Detection Rates in Experiment 1

	TX with 'Normal'	TX with 'Abormal'	Detection Rate
# of TX in Orig. Databset	1,221	18	100%
# of TX in Experiments	1,235	15	83.3%

The table denotes 100% of detecting anomaly in the original dataset, i.e. the training set with 'Normality' labels. This is because the given dataset was created with software-detected anomaly and human operator-detected anomaly detection. The sum of these two efforts often yields a 100% of determination on normality. The experiment with our machine learning framework yields 83.3% of detection, lower than 100%. This is because 83.3% of detection indicates the pure software-detected anomaly, none by human. As the result, it should be lower than 100%.

Experiment 2 is to evaluate the effectiveness of using *PAM*. We executed the system with *GAM* and with *PAM* respectively. The result is shown in Table 3.

Table 3. Detection rates in experiment 2

	TX with 'Normal'	TX with 'Abormal'	Detection Rate
Using GAM	1,239	11	61.1%
Using PAM	1,235	15	83.3%

The difference between the two cases is 22.2%, which is considerably high. It was shown that *PAM* specifies anomaly patterns which are more specific to users' behavior and patterns.

Experiment 3 is to evaluate the effectiveness of the activity 4. We executed the system by applying only the first 3 steps. And, we also executed it by applying all 4 activities. The result is shown in Table 4.

Table 4. Detection rates in experiment 3

	TX with 'Normal'	TX with 'Abormal'	Detection Rate
Applying Activities 1,2,3	1,238	13	72.2%
Applying All 4 Activities	1,235	15	83.3%

The difference between the two cases is 11.1%, showing the activity 4 of personalizing anomaly model for users' transactions behavior helps in increasing the detection rate.

Experiment 4 is to evaluate the effectiveness of *adaptiveness* of the proposed methodology. Recall that the adaptiveness is achieved with 2 features; personalizing anomaly model and refining the anomaly models inclemently. For this, we have run the system twice; 1st run with GAM and only the first 3 activities, and the 2nd run with PAM and all 4 activities. The result is shown in Table 5.

Table 5. Detection rates in experiment 4

	TX with 'Normal'	TX with 'Abormal'	Detection Rate
GAM with Activities 1,2,3	1,238	13	52.8%
PAM with All 4 Activities	1,235	15	83.3%

The difference between the two results is 30.5%, showing the adaptiveness of the framework is highly effective.

Through the four experiments, we can derive a comparison chart as in Figure 5.

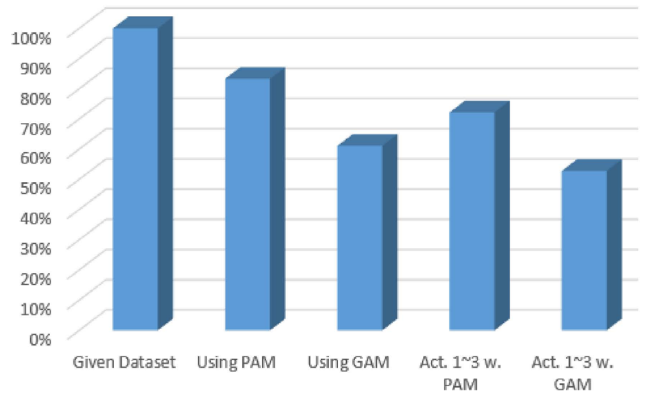


Figure 5. Comparision of 5 cases

Beside the given original dataset, anomaly detection with PAM with all four activities yields the highest accuracy. Running only the activities 1, 2, and 3 with GAM yields the lowest detection rate, indicating the 'adaptiveness of the anomaly model' is the essential factor in determining the overall detection rate.

6 Conclusion

FinTech has attracted attentions from industries, and many companies have been releasing FinTech services. One of the most important issues to be addressed in developing FitTech services is to keep users' transaction data secure. Due to the FinTech-specific characteristics such as diversity and evolvability in transaction models, managing anomalies of Fintech services is much more challenging. To address the characteristics, anomaly management of FinTech services requires a new level of effectiveness, intelligence, and timeliness in managing the anomaly of FinTech transactions. Hence, we propose a FinTech anomaly management approach by utilizing machine learning-based intelligence in deriving anomaly detection model and adaptive FinTech security provision. First, we presented a conceptual framework for efficiently managing FinTech anomalies including the overall process and the formal representation of key artifacts. And, we defined the algorithms of the process, which utilize the machine learning capability and adaptive nature of methods. With the proposed framework, it is feasible to manage FinTech anomalies in a timely and adaptive manner.

Acknowledgements

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIP) (2015R1A2A2A01004078).

References

[1] P. Schueffel, Taming the Best: A Scientific Definition of Fintech, *Journal of Innovation Management*, Vol. 4, No. 4, pp. 32-54, October, 2016.

[2] A. Adaballah, M. A. Maarof, A. Zainal, Fraud Detection System: A Survey, *Journal of Network and Computer Applications*, Vol. 68, pp. 90-113, June, 2016.

[3] M. Ahmed, A. N. Mahmood, M. R. Islam, A Survey of Anomaly Detection Techniques in Financial Domain, *Future Generation Computer Systems*, Vol. 55, pp. 278-288, February, 2016.

[4] N. A. L. Khac, M-T. Kechadi, Application of Data Mining for Anti-Money Laundering Detection: A Case Study, *2010 IEEE International Conference on Data Mining Workshops*, Sydney, Australia, 2010, pp. 577-584.

[5] W.-H. Chang, J.-S. Chang, Using Clustering Techniques to Analyze Fraudulent Behavior Changes in Online Auctions, *2010 International Conference on Networking and Information Technology*, Manila, Philippines, 2010, pp. 34-38.

[6] F. H. Glancy, S. B. Yadav, A Computation Model for Financial Reporting Fraud Detection, *Decision Support System*, Vol. 50, No. 3, pp. 595-601, February, 2011.

[7] L. Torgo, E. Lopes, Utility-based Fraud Detection, *22th International Joint Conference on Artificial Intelligence*, Barcelona, Spain, 2011, pp. 1517-1522.

[8] J. West, M. Bhattacharya, Intelligent Financial Fraud Detection: A Comprehensive Review, *Computers & Security*, Vol. 57, pp. 47-66, March, 2016.

[9] S. Bhattacharyya, S. Jha, K. Tharakunnel, J. C. Westland, Data Mining for Credit Card Fraud: A Comparative Study, *Decision Support Systems*, Vol. 50, No. 3, pp. 602-613, February, 2011.

[10] J.-S. Chang, W.-H. Chang, A Cost-Effective Method for Early Fraud Detection in Online Auctions, *10th International Conference on ICT and Knowledge Engineering*, Bangkok, Thailand, 2012, pp. 182-188.

[11] T. K. Behera, S. Panigrahi, Credit Card Fraud Detection: A Hybrid Approach using Fuzzy Clustering and Neural Network, *2nd International Conference on Advances in Computing and Communication Engineering*, Dehradun, India, 2015, pp. 494-499.

[12] Y. Sahin, E. Duman, Detecting Credit Card Fraud by ANN and Logistic Regression, *2011 International Symposium on Innovations in Intelligent Systems and Applications*, Istanbul, Turkey, 2011, pp. 315-319.

[13] S. Yaram, Machine Learning Algorithms for Document Clustering and Fraud Detection, *IEEE 2016 International Conference on Data Science and Engineering*, Kochin, India, 2016, pp. 1-6.

[14] S.O. Friend, R. Cooper, H. Finlay, Standard Deviation: The New Standard for Out-of-Pattern Transaction Analysis, *ACAMS Today*, Vol. 8, No. 1, January/February, 2009.

[15] V. Chandola, A. Banerjee, V. Kumar, Anomaly Detection: A

Survey, *ACM Computing Surveys*, Vol. 41, No. 3, Article 15, July, 2009.

[16] V. J. Hodge, J. Austin, A Survey of Outlier Detection Methodologies, *Artificial Intelligence Review*, Vol. 22, No. 2, pp. 85-126, October, 2004.

Biographies

Hyun Jung La is a software engineer at Workday, Inc., in California, USA. She received her Ph.D. in Computer Science at Soongsil University. Her research interests include platform software architecture and autonomous systems with machine learning analytics.

Soo Dong Kim is a professor in the School of Software at Soongsil University in Seoul, Korea. He received his Ph.D. in Computer Science at the University of Iowa, USA. His research interests include advanced software architecture and intelligent systems with machine learning.