Data Analytics to Detect Evolving Money Laundering

Murad Mehmet, Duminda Wijesekera
George Mason University
mmehmet@gmu.edu , dwijesek@gmu.edu

Abstract— **Money laundering evolves using multiple layers of trade, multi trading methods and uses multiple components in order to evade detection and prevention techniques. Consequently, detecting money laundering requires an analytical framework that can handle large amounts of unstructured, semi-structured and transactional data that stream at transactional speeds to detect business-complexities, and discover deliberately concealed relationships. Based on our prior work and a static risk model proposed in the Bank Security Act, we propose a dynamic risk model that assigns a risk score for every transaction being a potential component of a larger money-laundering scheme. We use social networks to connect missing links in such potential transaction sequences. Taken together we can provide a financial sector independent risk assessment to submitted transactions. The proposed risk model is validated using data from realistic scenarios and our already developed money laundering evolution detection framework (MLEDF) that we developed earlier using sequence matching, case-based analysis, social networks, and complex event processing to link fraudulent transaction trails. MLEDF has components to collect data, run them against business rules and evolution models, run detection algorithms and use social network analysis to connect potential participants.**

Keyword: Data Analytics; Social network analysis; Anti Money Laundering; Dynamic Risk Model ; Money laundering Risk .

I. INTRODUCTION

Money laundering (ML) is a major issue for the Department of Homeland Security (DHS) and US Treasury. Powered by modern tools, money launderers use complex schemes to avoid being detected by anti-money laundering (AML) systems. They dynamically evolve, expand and contract over fraudster networks in different countries. Social Network Analysis (SNA) techniques [1] are used by government agencies to track terrorist activities and networks. Because terrorist financing heavily depends on ML [2], any AML system must incorporate SNA to obtain reliable results. Schwartz [3] proposes a model to find criminal networks using social network analysis, building upon Borgatti's SNA-based key player approach [3]. One drawback of Borgatti's model is the failure to assign weights to actors and actor-actor relationships. In the recent past, we have developed algorithms incorporates Borgatti's SNA techniques with different weights' to social and business relationships to help complete missing links in potential money laundering chains.

The Financial Action Task Force (FATF) provides a static risk assessment of ML [4] strategies to examine ML related predicate crimes and known weaknesses of anti-money laundering (AML) systems. The Wolfsberg Group, made up of eleven leading international banks established standards, guidelines and a discretionary risk model [5] to counter money laundering. Both FATF and Wolfsberg Group say that monitoring customers is an essential part of countering money laundering and suggest that risks can be measured using metrics such as "Country risk", "Customer risk", and "Services risk", and leave weights assigned to each of these categories at the discretion of the evaluating organization. Based on these guidelines, banks use a quantitative model to evaluate transactional risk using attributes such as "Customer profile", "Product/service profile", and "Geographic profile".

The static risk model developed by Scor [6] accepts ML to be determined by "Agility" of adopting new rules per customer, "Complexity" of transactions and the "Secrecy" of transactional information and customer account" [6], but fails to assess other factors such as relationship networks, and dynamically changing factors. Kount [7] developed a dynamic scoring service to continuously monitor indicators of fraudulent credit card activity, and alert merchants of approved transactions that are linked to suspicious purchasing activities, that usually occur after identity theft. These suspicious purchases refer are transactions patterns that have never been witnessed before, such as the purchase of video games by senior citizen.

The rest of the paper is organized as follows. Section 2 describes the Money Laundering Evolution Detection Framework (MLEDF). Section 3 describes the SNA algorithm. Section 4 describes the dynamic risk model. Section 5 evaluates the performance of the MLEDF and dynamic risk model using real-life cases. Section 6 concludes the paper.

II. ML EVOLUTION DETECTION FRAMEWORK (MLEDF)

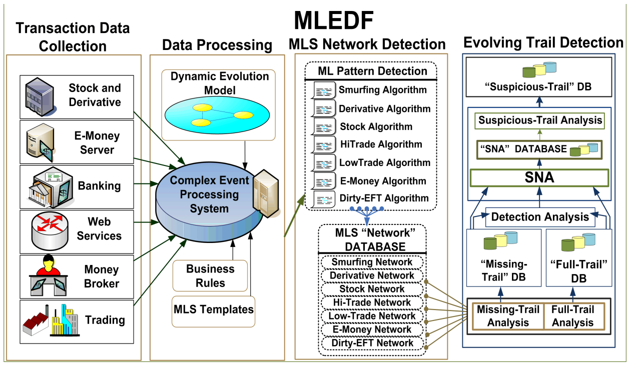


Fig. 1. Money Laundering Evolution Detection Framework

First we briefly describe how MLEDF works [8]. Obtaining data streams from multiple sources listed in the left hand column of Fig.1, using a complex event processing system. MLEDF uses four phases where output from one phase is used by the following phase and shown in the columns of Figure1.

(1) *Collecting Transactional Data*: Transaction processors or data collectors from Automated Clearing House such as EPN, FEDWIRE, and CHIPS send their data belonging to trade sectors such as Banking, Stock market, Derivative market, Web

based Services, Trading, Electronic Money, and Money Brokering. Relevant information is extracted form this data and used with transaction-independent data such as the economic status of the country, stock sales trends and the stock values during the day.

(2) *Data Processing*: Well-known MLS are identified and relevant attributes are collected from input data streams and submitted to our detection algorithms. The extracted data associated with each MLS pattern assigned to a specific MLS type using the following: (I) Business Rules: MLS business rules and red flags associated with each pattern, the rules associated with specific sector are used by the MLS detection algorithms to identify the MLS patterns. (II) MLS Template: Well-known MLS templates will be used during this phase. Currently, the templates have seven major pattern types with their different subtype combinations. This acts as a repository of known MLS. If a new form of MLS is discovered, then it will be added to this Database. (III) ML Evolution Model: Determines if the evolution of MLS is within the accepted trend of our model [8].

(3) *Detecting MLS Networks*: We use six algorithm (one for each) to detect Smurfing, Trade, Stock, Derivative, E-Money, and Dirty Electronic Funds Transfer (Dirty-EFT) schemes. Each algorithm uses a different method to capture the network associated with the specific type of MLS and in real-time output the discovered networks associated with the specific MLS pattern into a different database. Then, the discovered networks are reformatted and saved in a single database referred to as the "Network" Database to facilitate efficient analysis of the links among MLS networks.

(4) *ML Trail Analysis and Evolution Detection*: Four separate algorithms are run to find the "*Full-Trails*" [8], "*Missing-Trails*", and "*Suspicious-Trails*" of MLS networks. "*Full-Trail*" is a concatenated sequence of related schemes (MLS) act by itself to transfer money from one MLS to another until it reaches the final MLS, of which the orchestrator (i.e. the money launderer) is referred to as the "*EndBoss*". Similarly the orchestrator of the first scheme is referred to as the "*StartBoss*". "*Associates*" are other people involved in the sequence of fraud. "*Missing-Trail*" is a short Full-Trail that does not exceed have more than three related MLSs. A sample output from the Full Train Algorithm is given in Table 1, where a network ID (assigned by our detection algorithm), the duration of the laundering activity, if the money was withdrawn after the third transaction, the amount of money and the detected Start Boss and the detected End Boss are provided. We assume that the Missing-Trail is a premature Full-Trail with broken parts and missing links or evidence. A "*Suspicious-Trail*" is a combination of discovered Full-Trails and/or Missing-Trails constructed using algorithms that incorporate SNA and numerical analysis techniques. The algorithm "Detection Analysis" determines the evolution of the "*Full-Trail*"s such as the change to the number of involved associates, the changes to the cost of laundering, and changes to the laundering locations.

III. Social Network Analysis in MLEDF

In many cases money launderers intentionally obfuscate the money trail either by hiding it (for example by increasing the number of transactions and reducing the transacted amount), or using an unreported method such as a Hawala [15] (an honor based exchange system without records). As a solution, we use a social network among money launderers to link MLS trails when evidence of linkage is missing among transactions.

A. *Using Complex Event Processing in the SNA Module*

The critical question of ML experts is "*How fast and how well can we relate the different events in this universe of detected MLS?*" Using the Complex Event Processing (CEP) system StreamBase, we developed an algorithm to create chains of related MLSs where social or professional relations are used to transfer a fund to the next MLS until it reaches the final destination where the *End Boss* withdraws the money. If we modeled all of the chains as a separately and link them we run into a scalability issue in associating the multitude of different events of various MLSs. As a solution, we model each detected MLS as an event, and have various patterns of events categorized under six different types of MLS. For example, Full-Trail algorithm outputs a trail by using the functionality of CEP of perceiving the MLSs as a set of events. Without the CEP the MLS should dissolve into the constituent transactions to be analyzed and linked with other transactions from another MLS, consumes processing time and resources. The CEP can link MLSs, perceived as events, using various criterions without the need to add more complex sub-algorithms for each criterion. That is, the Full-Trail connects the dots that exist, but it is harder and slower to connect them without CEP capabilities. Full-Trail captures the trail in cases where all evidence are available, whereas the Suspicious-Trail attempts to construct the path where some edges along the path is missing.

B. *Integrating the "SNA" Module into MLEDF*

The "*Suspicious-Trail*" module is used to detect components of an actual "Full-Trail" even if there is a missing piece of evidence. This module investigates all available trails (Full-Trail and Missing-Trail) by using our SNA DB that contains the weights of relationships among MLS participants in order to determine if two trails are related by considering some attributes such as the amount of funds involved, location, affinity of participants, time, and methods used for laundering. Hence, the "*Suspicious-Trail*" module uses the "*SNA*" module to produce a new trail containing two or more trails that are related based on SNA even when we have not captured a transaction joining them or any other evidence. The new trail is created after making a calculation based on (SNA) results of a possible relationship between two or more Full-Trails and Missing-Trails. The generated evolution patterns and strategies are collected into the "*Suspicious-Trail*" Database.

TABLE I. Sample Output of the Full-Trail (Min 4 MLS)

Networks	TrailID	Duration	Withdraw	Amount	StartBoss	EndBoss
4, 91, 98 ,..	1232	76 Days	No	723,234	Boss956	Boss 153
24, 315, ..	1208	89 Days	No	890,165	Boss 103	Boss 827
405,783, ..	9724	19 Days	No	200,230	Boss 284	Boss 725

C. *The Components and Output of the "SNA" Module*

The architecture of the social network algorithm is shown in Figure 2. The SNA module generates and continuously updates two databases as outputs. The "*SuspectWeight*" database contains the weight of all relations detected in the

MLEDF and the "*Relations*" database containing the time and the record of all detected relations among pairs in MLEDF as shown in the bottom of Figure 2. The relations we capture are: (1) *UniqueFullTrailBosses* creating "StartBoss"–"EndBoss" pairs of "Full-Trail"s. (2) *UniqueFullTrailAssociates* creating "Asscoiate"-"Asscoiate" pairs of "Full-Trail"s. (3) *UniqueMissingTrailAssociates* createing "Asscoiate"-"Asscoiate" pairs of "Missing-Trail"s. (4) *SchemaBosses* creating Hashes for "StartBoss"-"EndBoss" relations. (5) *SchemaAssociateBoss* creating Hashes for all detected "Boss-Associate" relations. (6) *SchemaAssociate* creates Hashes for all detected "Associate-Associate" relationships. This hash represents the combinations of relationships among the associates of the same MLS, even if they do not interact/transact with each other directly. (7) *Family* creating "Family" relation between lineage-wise related pairs. (8) *Business* creating business-wise related pairs. Each such relationship is shown in Figure 2. We compute these relationships and assign weights to them as shown in Algorithm 1 describe in Table II.

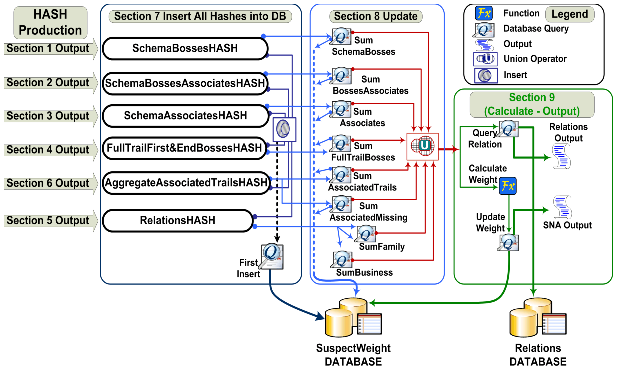


Figure 2: The Social Network Analysis Module

The relationship weights as assigned so that higher weight indicates more possible hidden interactions. Weights are calculated by adding parameters for each of the corresponding events as follows:

1. Each detected "MLS" weights of 10 will be added to start/end boss couple, 5 for each boss/associate combination, and 1 for each non-repeating associate/associate combination.
2. Each detected "Missing-Trail", 15 will be added to each associate non-repeating combination.
3. Each detected "Full-Trail" add 20 to each associate combination and 25 to the start and end boss.

Other strong relationships are also counted where "Family" ties will add 250 to the couple, and each "Business" relationship will add 250 to the couple. We chose the weights and, verified them in a limited engagement with a trusted third party (see Section V), but can be changed in Algorithm 1.

In the SNA Algorithm given in Table II, steps 1 and 2 define the hash function, and input and DBs constants associated with the different weights and the hash functions. Steps 3 and 4 create hashes for "*Boss-Boss*", "*Boss-Associate*", and "*Associate-Associate*" of MLSs. Steps 5 and 6 create the same hashes for Full-Trails. Steps 7 and 8 create hashes for Missing-Trails and special relations (of family and business). Step 9 computes the WeightOutput of a hash HRel. Sample Suspect Weights obtained from Algorithm 1 is shown in Table IV. This corresponds to Relationships given in Table III.

TABLE II. SOCIAL NETWORK ANALYSIS ALGORITHM

```
1 FUNCTION HASH (String1,String2){return concatenate(sort(En1,En2))};
2 INPUT MLS DetectedMLS; Relnship; MT MissingTrail; FT FullTrail; DB
  HRel ( Hash, #"Time", Type, Person1, Person2) KEY (Hash, #"Time",
  Type); DB SuspectWeightOutput ( hash, weight) KEY (hash);
3 UPDATE  HRel   SET MLSBoss++,  MLSAssocBoss++,
  MLSAssocBoss++    WHERE HRel.hash == HASH(MLS.sBoss,
  MLS.eBoss) , HASH(MLS.Assoc, MLS.eBoss) ,  HASH(MLS.Assoc,
  MLS.sBoss) and TypeMatch
4 FOR EACH (MLS.Assoc as Assoc1, MLS.Assoc as Assoc2) UPDATE
  HRel  SET MLSAssoc++ WHERE HRel.hash == HASH(MLS.Assoc1,
  MLS.Assoc2);
5 FOR EACH (FT.Assoc as Assoc1, FT.Assoc as Assoc2)  UPDATE HRel
  SET FTAssoc = FTAssoc++ WHERE H.hash ==HASH(Assoc1,Assoc2);
6 UPDATE HRel  SET FTBoss++ WHERE HRel.hash ==
  HASH(FT.sBoss, FT.eBoss);
7 FOR EACH (MTrail.Assoc as Assoc1, MTrail.Assoc as Assoc2)
  UPDATE HRel  SET  MTAssoc++ WHERE HRel.hash ==
  HASH(Assoc1, Assoc2) and TM;
8 UPDATE HRel  SET Business++, Family++ WHERE HRel.hash ==
  HASH   (Relnship.person1, Relnship.person2) AND Relnship.type ==
  "BUSINESS","FAMILY";
9 SELECT HRel.hash, (25*HRel.FTBoss +20*HRel.FTAssoc
   +15*HRel.MTAssoc + 1*HRel.MLSAssoc + 5*HRel.MLSAssocBoss
   +10*HRel.MLSBoss + 250*H.Business +250*H. Family)  as
  SuspectWeightOutput ;
```

TABLE III. SAMPLE SELECTION FROM OUTPUT OF "RELATIONS"

DetectTime	Hash	Type	Entity1	Entity2
2012121915	Comp10Comp8	FullTrailAssociates	Comp10	Comp8
2012121923	Comp10Comp5	SchemaBossAssociate	Comp10	Comp5
2012122005	Comp10Assoc7	MissingTrailAssociates	Comp10	Assoc7
2012122112	Assoc7Assoc5	SchemaAssociates	Assoc7	Assoc5
2012122214	Comp10EndBoss	SchemaBosses	Comp10	EndBoss
2012122220	StartBossEndBoss	FullTrailBosses	StartBoss	EndBoss

TABLE IV. SAMPLE SELECTION FROM OUTPUT OF "SUSPECTWEIGHT"

Weight	Hash	Weight	Hash	Weight	Hash
30	Comp10Comp8	10	Comp11Comp2	10	Assoc1Comp1
30	Comp10Comp7	10	Comp11EndBoss	30	Assoc1Comp4
15	Comp10Comp5	10	Comp1Comp2	35	Comp4Comp6
30	Comp10EndBoss	20	Comp6EndBoss	20	Assoc1Assoc5
10	StartBossEndBoss	20	Comp7Comp8	0	Assoc1Assoc9

IV. THE DYNAMIC RISK MODEL

Existing AML systems do not relate different products types, entities, and business lines involved in different combinations of complicated ML schemes. Industry specific AML systems use industry specific static risk models and therefore do not capture known dynamics of MLS evolutions. Countering ML and other forms of fraud requires industry-wide risk analysis method to where the risk score is updated dynamically and include transactional behavior related to the ML, such as the social relations and past associations with money laundering. Therefore, we create a dynamic risk model that incorporates the static attributes used by others, such as the senders and recipient's static profiles and dynamic social connection attributes of the transactions that we capture in our MLEDF system.

A. The Static Risk Model of Bnak Secrecy Act

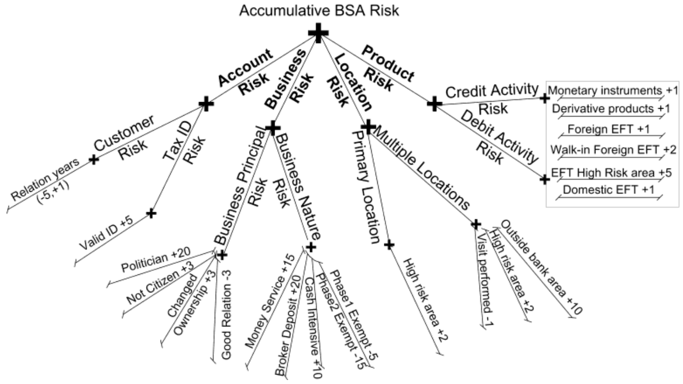


Figure 3. The enhanced BSA Static Risk Modeling [9]

The Currency and Financial Transactions Reporting Act (CFTRA) of 1970 later amended to counter money laundering and financial crimes [11, 12, 13] and again amended by Title III of the PATRIOT Act of 2001 and other legislations, and is now commonly referred to as the "Bank Security Act" (BSA) mandates banks to monitor transactions and maintain records of initial and periodic risk scores for customers. Their risk model identify and analyze specific “products and services”, “customers and entities”, and “geographical locations” and categorize them as “high", "medium", and "low", and add the risk rating of all categories to obtain the overall accumulative risk score. We enhanced the BSA inspired static risk with aggregated static risk to reflect changing dynamics of ML and its consequences on the static risk calculation shown in Figure 3. The risk rates assigned in Figure 3 are obtained from [9], with suggested enhancements in the upper right hand box. Definition 1 captures these attributes and scores.

Definition 1 [Local Static Risk Score (LSRS) and Risk Categories]: The Local Static Risk Score is the sum of the following attributes and their assignable integer values;

Account Risk Range:[-5,+10], *Location Risk* Range: [-1,+10]

Business Risk Range: [-15,+20], *Product Risk* Range: [0,+5]

Here *Account Risk* is the sum of *Customer Risk* [-5, +10] and *Tax ID Risk* [+5]. The *Location Risk* is the Sum of *Primary Location Risk* [+2]. The sum of the Risks of Non Primary Locations, where each *Non Primary Risk* is a value in the Range [-1,+10]. The *Business Risk* is defined as the sum of *Business Primary Risk* [-3, +20] and *Business Nature Risk* [-15, +20]. The *Product Risk* is the sum of *Debit Activity Risk* [0, +5] and *Credit Activity Risk*: [0, +5].

The BSA risk score is the sum of the component risk scores of Account Risk, Location Risk, Business Risk and Product Risk. Each of these components risks are also sums of further sub components as specified in Definition 1. Possible computed value for a customer is an integer value for between -23 and +20. The details of risks used in Definition1 are as follows. The Account Risk is the Risk due to customer’s reputation and a risk assigned due to providing / not providing a TAX ID. The Locations Risk is the sum of having multiple business Locations, and the risk associated with the Primary business Location. The Business risk is the sum of the risk due to the Principal Owner and the risk associated with the Nature of the Business. The financial product risk is associated with the debit and credit activities. We amended the factors of “product risk” in the BSA model to include a risk factor of the derivative market activity. We also reduced the risk weight of three factors in the “business risk” of BSA model from the original value of “+30” to the new value of “+20”, as the total risk score of “30” is the cut-off for an alert to the management of the financial institution. The reduction of the weight of the three factors to “+20” is necessary to lower the aggressiveness of the risk model. Definition 2 categorize these risks as Low, Medium, High and Extreme and are again an extension of the values in [9].

Definition 2 [Categorizing Local Static Risk Scores]: Local Static Risk Scores (LSRS) are categorized as low, medium, high and extremely high based on range of the totally calculated score: ***Low*** [-23, 4], ***Moderate*** [+5, +14], ***High*** [+15, +30], ***Extreme Risk*** [+31, +153].

B. Accumative Static Risk Score

To compute the risk of transacting customers, in addition to Static Local Risk Score (LSRS), risk of recent transactions need to be taken into account. We propose a simplified mechanism of exchanging aggregate risk scores assigned to customer transactions, because a running average may not expose all the data of all transactions and therefore may not violate privacy. Formally, let *TRN(O,R)*, be a transaction with originator *O* and recipient *R*, and let $TRN_A \equiv <TRN_{A1}(x_1,y_1), TRN_{A2}(x_2,y_2),, \ldots\ldots , TRN_{An}(x_n,y_n)>$, listed in newest to oldest transaction order represent the last **n** transactions of A. Let $Partner_i(A,TRN_{Ai}(x_i,y_i))$ represent the entity other than A and $<LSRS(Partner_1(A,TRN_{Ai}(x_i,y_i)))>$be the LSRS values of partners of A in the last n transactions. Then recursively define the Exponential Moving Average (EMA) risk as: EMA(i) = $\mathrm{LSRS}(\mathrm{Partner}_i(\mathrm{A},\mathrm{TRN}_{\mathrm{A}i}(x_i,y_i)))$* k + EMA(i-1) * (1 – k) where k = 2/(n+1) .

Definition 3 [Receiver’s/Originator’s Average Risk and Variance]: These averages are calculated by the bank that holds the account of entity A, it is done by calculating the exponential moving average of the LSRS of the last n transacting partners of A,

Let $\mathbf{EMA_i}$ *be* $LSRS(Partner_1(A,TRN_{A1}(xi,yi)))* k + EMA(i-1) * (1 – k)$ *where* $k = 2/(n+1)$, *and where* $A=x_i$ *for all* $i<n$.

Let $\boldsymbol{Var_A}$ *be* $LSRS_A– Average(LSRS(Partner_1(A,TRN_{A1}(x_1,y_1)))), \ldots, LSRS(Partner_n(A,TRN_{Ain}(x_n,y_n)))$.

As with the Average, $\mathrm{VAR}_{\mathrm{A}i}$ computes the receiver’s and originators risk based on the value of $\mathrm{Partner}_i(\mathrm{A},\mathrm{TRANA}_i(x_i,y_i))$. When $\mathrm{Partner}_i(\mathrm{A},\mathrm{TRANA}_i(x_i,y_i))$ = A= Receiver for all i<n. Then EMAi computes the receiver’s risk and When $\mathrm{Partner}_i(\mathrm{A},\mathrm{TRANA}_i(x_i,y_i))$ = A= Originator for all i<n. Then EMAi computes the originator’s risk. The *RA/OA* parameters assess the risks associated with the participation/involvement of an entity in the ML, by analyzing the affinity/role in the money-flow of a laundering process. The *RA* and *OA* used to calculate/keep a record of the historical activity and the divergence in pattern of receiving or sending funds. The pattern are used as an indicator for assessing a risk penalty, comparing the current *RA/OA* value with the value of 90 days and 180 days ago (*RA/OA*) indicate the transactional tendency of the entity.

We assign a penalty and reward system for entities so that an entity that continuously transacts in high and increasingly risky pattern is subject to penalties for having an increased LSRS, and vice versa. Thus our penalty and rewards system self-adjusts and the leverage provided by this self-adjustment avoid maintaining the risk value of an entity at a static level. The penalty can be set upon the needs of the financial institution and the regulations of the country, although optimal levels are shown in the formula below. The optimum penalty/reward are produced to allow entities to retain their old static risk levels in between one and **n** transactions. The criteria defined below indicate that if the aggregate risk sores is higher than 90 days ago which is higher than the same value 180 days ago this entity's transacting risk is on the increase.

Definition 4 [Penalties and Rewards]: We define $RA_{0M,}$ RA_{3M}, $RA_{6M,}$ OA_{0M}, OA_{3M}, and OA_{6M} to be respectively the current, three months, and six months old values of RA and OA value, for any entity. Let *RA-Inc*, *RA-Dec*, *OA-Inc* and *OA-Dec* be defined as $(RA_{6M}<RA_{3M}<RA_{0M})$, $(RA_{6M}>RA_{3M}>RA_{0M})$, $(OA_{6M}<OA_{3M}<OA_{0M})$, and $(OA_{6M}>OA_{3M}>OA_{0M})$ and Static Risk Penalty and Reward (SRPR) as:

$(RA\text{-}Inc)\wedge(OA\text{-}Inc)\wedge(RA\geq LSRS)\wedge(LSRS>35)\wedge(RV\geq 5)\wedge(OV\geq 5) \Rightarrow SRPR=+5$

$(RA\text{-}Inc)\wedge(OA\text{-}Dec)\wedge(RA\geq LSRS)\wedge(LSRS>35)\wedge(RV\geq 5)\wedge(OV\geq 5) \Rightarrow SRPR=+3$

$(RA\text{-}Dec)\wedge(OA\text{-}Inc)\wedge(RA\geq LSRS)\wedge(LSRS>35)\wedge(RV\geq 5)\wedge(OV\geq 5) \Rightarrow SRPR=+3$

$(RA\text{-}Dec)\wedge(OA\text{-}Dec)\wedge(RA\geq LSRS)\wedge(LSRS>35)\wedge(RV\geq 5)\wedge(OV\geq 5) \Rightarrow SRPR=+2$

$(RA\text{-}Dec)\wedge(OA\text{-}Dec)\wedge(RA\geq LSRS)\wedge(LSRS>35)\wedge(RV<5)\wedge(OV<5) \Rightarrow SRPR=-2$

$(RA\text{-}Dec)\wedge(OA\text{-}Dec)\wedge(RA\leq LSRS)\wedge(LSRS>35)\wedge(RV\geq 5)\wedge(OV\geq 5) \Rightarrow SRPR=-3$

$((RA\text{-}Any)\vee(OA\text{-}Any))\wedge (RA<LSRS) \Rightarrow SRPR=0$

A detailed rationale for this definition is described in [14]. The LSRS will be calculated every time the transaction occurs. For example, the first line of Definition 4 says that if conditions (1) "RA6<RA3<RA", (2) OA6<OA3<OA, (3) RA≥LSRS and (4) RV>OV>0 are met, the SRP of "+5" will be imposed. The RA is the primary factor that LSRS depends upon on to determine the penalty value due to the fact that receiving the funds is where the money laundering fraud starts. The penalty and reward point system will have the upper and lower bounds, in order to maintain the LSRS within its boundaries so that their accumulation will have a fix point (risk saturation point) in its decreasing or increasing trend. There is no need to apply the penalty on an entity that is in maximum risk levels of LSRS, as the purpose is to provide the transacting entity with the ability to reduce the risk.

Definition 5 [Accumulative Static Risk Score (ASRS)]: of an entity is the sum of the local static risk score and static risk penalty and reward. Thus, ASRS =LSRS + SRPR.

C. *Accumulative Dynamic Risk Score*

The dynamics of none-static risk scoring was designed considering the following criteria: (1) *Continuous scoring*: The score is calculated per every transaction. (2) *Automatic scoring*: Risk computation does not require the involvement of an expert. (3) *Correlation of past transactions*: Risk score correlate transactions with current one.

We have developed an algorithm to assigning weights to relations, the so-called Dynamic Relation Extract Algorithm (DREA) [14] that searches the SNA DB "SuspectWeight" for the detected past **n** ML activities of the entity A. The algorithm is similar to the algorithm used in the "SNA" module of MLEDF as explained in Section 2. Using this algorithm, we assign a risk weight for entity A, for each detected ML activity in the SNA DB "SuspectWeight" by adding parameters for each of the corresponding events resulting in the accumulative risk weight as follows:_(1)_For each detected MLS, add 5 to start/end boss couple, 2 for each boss/associate combination, and 1 for each associate/associate non-repeating combination. For each Missing-Trail add 3 to each associate non-repeating combination. The Full-Trails adds 3 to each associate combination and 10 to the start/end boss. Again these are our sample values that can be changed by any institutions. We omit an algorithmic presentation due to lack of space.

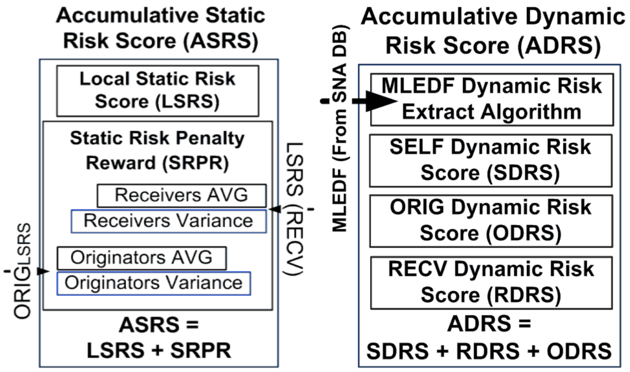

Accumulative Static Risk Score (ASRS)
Local Static Risk Score (LSRS)
Static Risk Penalty Reward (SRPR)
Receivers AVG
Receivers Variance
Originators AVG
Originators Variance
ORIG-LSRS
LSRS (RECV)
ASRS = LSRS + SRPR
MLEDF (From SNA DB)
Accumulative Dynamic Risk Score (ADRS)
MLEDF Dynamic Risk Extract Algorithm
SELF Dynamic Risk Score (SDRS)
ORIG Dynamic Risk Score (ODRS)
RECV Dynamic Risk Score (RDRS)
ADRS = SDRS + RDRS + ODRS

Figure 4. The Two Componenets of ML Dynamic Risk Model

We also compute a risk score named the Self Adjusting Dynamic Risk Score (SDRS) that assigns a risk weight to the transactional history of transacting entity (say) A in the database "Suspect-Weight" in the DREA algorithm that we refer to as DREA(Entity A)].

Definition 6 [Receivers' / Originators' Dynamic Risk Score (RDRS/ODRS)]: Calculates the aggregate risk weight, based on the relations history of the last **n** entities ($R_{1,\ldots,}R_n$) funds receiving from, and the last n entities ($O_{1,\ldots,}O_n$) funds originating to the entity A. The average weight of receiving /originating entities is obtained by calculating the average of $DREA(R_1)$, .. ,$DREA(R_n)$ and $DREA(O_1)$, .. ,$DREA(O_n)$. We produce ODRS and RDRS.

Definition 7 [Accumulative Dynamic Risk Score (ADRS)]: Of an entity is the sum of SDRS, RDRS and ODRS. That is ADRS = SDR+ RDR+ODR.

D. *Accumlative Transaction Scoring Based on Dyanmic Risk*

Static and dynamic risks are correlated to the analytics of transaction scoring, in order to identify transactions with high-risk score pertaining to ML, and to prevent transaction sequences from being executed. The correlations used in the dynamic risk scoring can be used to detect and track transactions belong of ML schemes.

Definition 8 [Accumulative Transaction Score (ATS)]:

The ATS is calculated as the average risk of (ADRS, ASRS, LSRS) of the two transacting entities.
Receiver-ATS=∑Receiver(ADRS,ASRS),
Originator-ATS = ∑ Originator (ADRS, ASRS).
ATS = AVG (Receiver-ATS , Originator-ATS).

Definition 9: Comprehensive Transaction Data (CTD): The triple (LSRS, ASRS, ADRS) is said to be the comprehensive transaction data (CTD*).*

Thus, our comprehensive dynamic risk model consists of two parts, computing the static risk, as by amending the BSA risk model [9] and computing the dynamic risk per every transaction. Figure 4 summarizes the two aspects of our risk model and the data used compute the individual components. As the figure shows, the static risk score is summarized in the accumulative static risk score (an enhancement of the BSA model) and the Dynamic risk score that takes the originators and recipients running averages of static risk scores and other properties of the transactions and SNA information to compute a dynamic risk value per each transaction. As stated, this value is fed back to the running averages and variances of this static risk scores. This latter step requires the financial institutions to share such risk estimates along with transactions.

V. Validation

We used sanitized real-life cases to test and validate the dynamic risk model with MLEDF and transaction scoring. Our case studies are based on data provided from an organization we refer as Trusted Third Party (TTP), which is authorized to collect information and track records of financial exchanges.

A. *Experimental Evaluation and Valiation of MLEDF*

We introduced a three phase testing prototype to examine MLEDF and detection algorithms. All three phases focused on testing and validating the components of MLS, Full-Trails, and Suspicious-Trails. The first phase focused on testing all components and the other tests focus on Full-Trail and Suspicious-Trail components.

Test without noise: This test is designed to test every module of MLEDF, including detection algorithms and trail analysis modules. These tests evaluate the false positive rates (FPR) and false negative rates (FNR) by comparing the results of the test with the data feed that contains the patterns of single MLS, pair of MLSs, and Full-Trails. The desired result was to have a list of the validation result identical to the list in the data feed. We tested the efficiency to keep up with the speed of the data feed by using the time window feature in the StreamBase [10]. By setting the time window to glide over only one event at a time tick in the StreamBase system, we made the detection algorithms to run at the normal speed of one event at one time tick. By design, an algorithm that cannot attain the speed of event production will not be able to capture MLS events or the Full-Trail, thereby generating false negatives.

Each of the six detection algorithms were tested with their own data feeds in order to verify that we were able to detect a single event MLS without false positives and false negatives. The algorithm-specific dataset feed was generated using the built in feed generator working with our pattern specific event generator. Afterwards, we tested the "Missing-Trail" by feeding linked pairs of MLSs into the MLEDF. The linked/related pairs are randomly selected from the set of six types of MLS. As mentioned, any pair of linked MLS will make it to "Missing-Trail" and not into "Full-Trail", due to the required depth. Finally, we tested the detection and evolution of "Full-Trail"s by feeding trails generated from various laundering strategies used in our sample real-life cases.

The process of creating the "Full-Trail" started with creating an MLS type out of the six MLS types of Smurfing, Trading, DirtyEFT, Stock, Derivative, E-Money. Once the selection of first MLS is made, we create ta series of linked MLS based on conditions such as geography, amount of money, time, complexity of the schema and difficulty of tracking. The trails were created using different criteria and randomizing them using a normal distribution. We created the Full-Trail feeds using the generator to not exceed 10 levels of depth of linked MLSs. These trails were either a variant or a subsection of one of the real-life cases that were similar in terms of complexity and participants.

At the normal speed of one event at one time tick of the CEP system, the test result in zero false positive rates and false negative rates. It is highly improbable to get a false positive trail due to the business rules that define them, and due to the accuracy and granular level of linking transactions. We did not get any false positive rate (FPR) or false negative rate (FNR) in the MLS tests due to the synthetic nature of the data. When we increased the speed of the data generated to 10 times and 100 times the normal speed, we observed a FPR and FNR in the objects detected in the Full-Trail algorithms. Increasing the speed of processing did not produce FPR and NFR for a single MLS, but it produced FPR and FNR for MLS pairs at speeds that were multiples of 100s. The term "object" in this graph refers to the three different patterns of single MLS, pair MLS, and Full-Trail in the proprietary test of the specific object (Object in the first pattern tests to the first pattern single MLS, in the second to MLS pair, in the third to Full-Trail). The values of FRP and FNR reflect the number of falsely detected objects.

Test with subtle noise: This is the most relevant accuracy test of our detection algorithms. The goal of this test was to mislead the detection algorithms by generating false positives and false negatives synthetic data. The test had three separate phases: injecting the scheme participants, injecting subtle transactions, and inserting similar MLSs. A subtle transaction is a transaction with ±5% of an actual transaction amount in a MLS. A similar MLS is identical to a real-life MLS with the same set of participants but with the MLS value is ±10% of the laundered amount of the actual MLS. The injection speed was set to normal processing speed, 10 times faster, and 100 times faster. The test of injecting transactions and MLSs is setup considering each MLS type. For example, in the test of smurfing, we created only smurfing MLS and smurfing transactions that can extend vertically up to 20 levels of depth and horizontally to 30 levels of depth. When we were generating the MLSs our measures did vary based upon the MLS. We did not use artificially created none-real life cases. For example, we did not use a smurfing MLS with 100 levels deep, as that is uncommon and impractical to launder money. We also did not inject other types of MLSs into the injection test of a specific MLS. However, in the Full-Trail test, we injected all types of MLS because by design, a Full-Trail is required to have different types of MLS under the same Full-Trail.

The test produced low FNR and low FPR for transaction and MLS injection when the phases were executed at normal processing speed. Those rates increased in the phases when tests were executed at faster processing speed. One way to imitate the data rate of real production environment is to run the CEP tests at a faster rate, thereby overloading the system with processing and analytics while attempting to keep pace with the data stream. The goal was to evaluate the effectiveness of "Full-Trail" detection when the system absorbs data at a higher rate while performing the analysis. Due to the design methodology of detection algorithms and the complexity of the business rules of MLS detection, their false detection rates stayed at low levels (less than 5%) even with injection similar transactions and MLSs, at a higher data-feed speed (1000% and 10000% speed).

Meeting the design principles, the "Full-Trail" and "Suspicious-Trail" results remained at low rates for both false positive and false negative. Therefore, all the subtle single MLS created with injected data ended in the "Missing-Trail", where they did not exceed the depth of 3 consecutive MLSs. Some reasons for this success in trail analysis and avoiding any negative impact are (1) MLEDF is designed in a strict and granular method, especially for matching MLSs within trails, (2) SNA is used in the trail analysis algorithms, (3) Adopted the criterion to follow the direction of the flow of the laundered-money. MLS is not expected to terminate with funds remaining in the account. The money must flow in some direction in order to be laundered, or must be withdrawn by the launderer. The Figures 5, 6 and 7 show the results of the number (quantity) of the transactions resulted in false positive and false negative, as explained in the previous paragraph. The figures show the number of FP and FN patterns of each phase from the three injection phases of the Test II, along with the results from running the test at different speed (1000% and 10000% speed).

Test with longer synthetic full-trails: This was the hardest level of performance testing of the system and accuracy-testing of the detection algorithms we carried out. In this test, the dataset was permutated over a repository of different real-life cases. Afterwards, the dataset was combined with randomized MLS to generate deep vertical levels of "Full-Trail"s and "Suspicious-Trail"s. The randomization followed the same principles used in Test II's injection testing. The test was designed to assess the performance of MLEDF in capturing real-life data and analyzing them on the fly. The desired test result was to generate low FPR and FNR. The test module generated all synthetic data from real-life cases and tests were similar to real-life scenarios, considering that there are limited ways to manipulate a MLS. The test program functions as follows: (1) Set a trail depth. The program enters a loop and builds a trail by choosing a first scheme from of each MLS type at random, as it was described in Test I in building the Full-Trails. (2) The loop continues by creating an MLS that can be linked by funds, time, location and complexity to the current MLS. We repeated the step above with the exception of not creating any Smurfing MLS for the rest of the levels. (3) The permutation continues until the system reached the last level, where we always choose an MLS of type DirtyEFT with a withdrawal in order to generate the trail termination point, as by definition a trail will end with the withdrawal of money. (4) The test were repeated the process of trail generation forever at the maximum possible speed. (5) The testing module saved the arrival time of the last DirtyEFT and subtracted that from the build times of the trail, thereby obtaining Milliseconds difference in trail processing times.

Our data was generated for worst-case scenarios to ensure that they are more complex and the performance was evaluated only in most resource consuming cases. Displayed results represent the performance of data generated without any repetitive bosses or associates. Hence, the dataset consumes a significant number of resources.

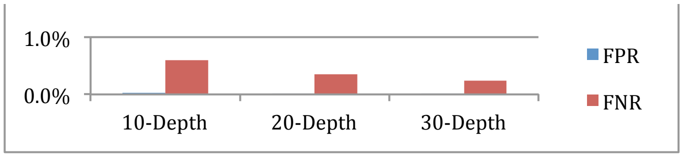
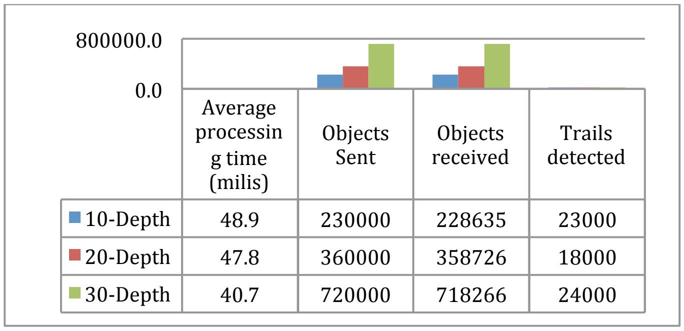


Figure 5. False Positive and False Negative Percentages of Test III



	Average processin g time (milis)	Objects Sent	Objects received	Trails detected
10-Depth	48.9	230000	228635	23000
20-Depth	47.8	360000	358726	18000
30-Depth	40.7	720000	718266	24000

Figure 6. Number of Detected Trails in Test III for Faster Data Rates

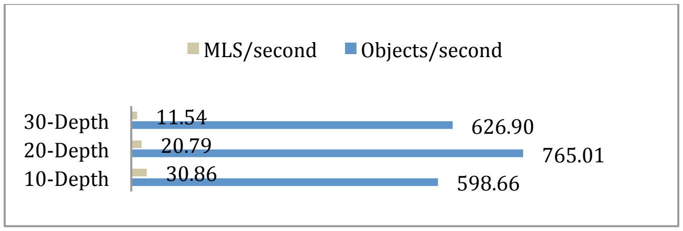


Figure 7. Pattern Generation Speed for Test III

B. *Experimental Evaluation and Validation of Money Laundering Dynamic Risk Model*

Test Methodology: We introduced a four phase risk model testing prototype to examine the three different versions of static risk model, and the dynamic risk model: (1) T1: Using standard average instead of the exponential average in calculating static risk. (2) T2: Using exponential average, but applying only penalty and no reward in calculating static risk. (3) T3: Using exponential average, apply both penalty and reward in calculating static risk. (4) T4: Using detected schemes, from the output of MLEDF, to produce dynamic risk scores.

Injected Data Phases: Four different types of transactions were injected in each of test phases with LSRS value of (10, 20, 30, 40) in the transactions of each test.

Test Goals: (1) Produce risk levels above certain threshold for continuously riskily transacting entities. (2) Effectiveness when certain patterns (all high risk or all low risk) were injected, (A) Does the ADRS/ASRS saturates at some fixedpoint level? (B) False Positive (FP): ADRS/ASRS continue to grow towards the high risk level of the continuously injected data (C) False Negative (FN): ADRS/ASRS deviates towards the low risk level of the continuously injected data (D) Maintain a desired risk level for bad entities even if they deliberately transact with good entities, in order to lower their risk profile.

Results: The dynamic risk model (T4) produces FP for transactions of none-MLEDF entities. The rate was less than 5% and that was satisfactory considering the large amount of transactions. This is advantageous compared with risk models that do not assess the risk of being involved in MLS, considering the factors of increasing risk scores of MLEDF entities.

Validation Statement: We used the StreamBase Studio [10] platform in each test of (T1, T2, T3, T4) and with each of the four data injection phases (by only injecting entities did not exist in MLEDF). The false negative rate was below 1% in phase 1 of all tests, and 0% in remaining three phases of data injection for all tests. The false positive rate was below 5% for T4, and lesser for other the three static tests (T1, T2, T3). In test T4 and with each of the four data injection phases by injecting entities did exist in MLEDF. The false negative rate for T4 (When only injecting entities that are already detected by MLEDF) is the highest at 11% when entities with high static risk (of LSRS 30) are injected in phase 4, then at 9% in phase 1 when high static risk score (of LSRS 25) are injected, then at 8% in phase 3, and finally at 3% in phase2 when low risk score (of LSRS 10) is injected. The false negative rate is 0% in all phases of test T4. Table V summarizes our findings. Figure 9 shows the false positive and false negative rates for injecting MLS's with 10, 20, 25 and 30 LSRS values.

TABLE V. NUMBER OF TRANSACTIONS WITH FN AND FP RISK

Transaction Injection/Test Type	T1	T2	T3	T4
Total Generated Transactions	240387	240387	240387	240387
Originators not from MLEDF	227	227	227	227
Originators from MLEDF	59	59	59	59
Unique Receivers	936	936	936	936
Injected MLEDF Transactions	9851	9851	9851	9851
FP- Phase1- Growing Risk	9	17	14	26
FN- Phase1- Declining Risk	0	0	2	0
FP- Phase2- Growing Risk	2	8	3	12
FN- Phase2- Declining Risk	6	1	2	0
FP- Phase3- Growing Risk	7	14	10	21
FN- Phase3- Declining Risk	0	0	0	0
FP- Phase4- Growing Risk	14	28	20	44
FN -Phase4- Declining Risk	0	0	0	0

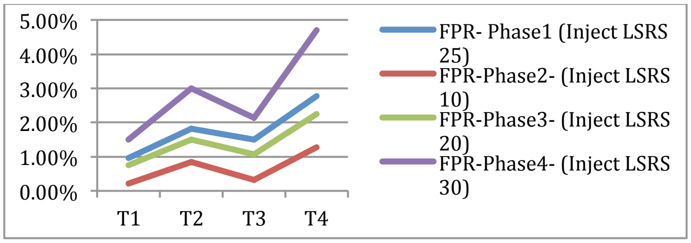


Figure 8. False Positive Rate for each risk models after data injection (none-MLEDF Entities)

VI. CONCLUSIONS

We implemented a multiphase, multilevel, and multi-component methodology to detect evolving money-laundering schemes using known methods, influenced by economic factors. We have created a framework to detect the evolution of MLS and implemented a system to include SNA for detecting and linking related ML networks. This linkage will function properly even when all evidence is unavailable. We defined the choreographies that could be used to detect the evolution of the sophisticated MLS. We have shown how to detect and capture the evolving and complex trails of MLS using SB.

We enhanced the BSA inspired static risk with aggregated static risk, to reflect the changing dynamics of the ML and its consequences on the risk calculation. Our risk model factors in the initial account-opening risk as well as subsequent transactional risks, and it presents a risk score that is valid within and outside the boundaries of a single financial institution. We extended the static risk model to develop a MLEDF-dependent risk modeling, in order to produce a comprehensive ML risk modeling in combination with the aggregated static risk model. The aggregated static risk will be completed with integration of the MLEDF-dependent risk modeling, which captures the hidden, and dynamic, relations among none-transacted entities. Such a risk model is used to create a valid and accurate transaction scoring system to be used in a ML prevention system.

REFERENCES

[1] C. Weinstein, W. Campbell, B. Delaney, G. O'Leary, "Modeling and detection techniques for Counter-Terror Social Network Analysis and Intent Recognition," Aerospace conference, IEEE, 2009.

[2] Financial Action Task Force, "Global Money Laundering & Terrorist Financing Threat Assessment Annual Report," February 2013.

[3] D. Schwartz and T. Rouselle, "Using social network analysis to target criminal networks," Trends in Organized Crime, 2008.

[4] Financial Action Task Force, "Money Laundering & Terrorist Financing Risk Assessment Strategies," June 2008.

[5] Wolfsberg Group, "Guidance on a Risk Based Approach - Wolfsberg Principles", 2006.

[6] Scor Inc, "The risk of money laundering: Prevention, challenges, outlook", 2008.

[7] Kount Inc, "Dynamic Scoring and Rescoring", 2011.

[8] M. Mehmet and D. Wijesekera, "Detecting the Evolution of Money Laundering Schemes," IFIP WG 11.9 Conf. on Digital Forensics, 2013.

[9] BankersOnline, "Risk Rating - Commercial Risk Rating Spread-sheet", http://www.bankersonline.com/tools/bc_commercialriskrating.xls

[10] StreamBase, 'Powerful Real-Time Architecture for Today's High Performance Modern Intelligence Systems', Federal Government, Defense, and Intelligence Applications, 2012.

[11] FinCEN (2013), *Answers to Frequently Asked Bank Secrecy Act Questions*, accessible via http://www.fincen.gov/statutes_regs/guidance/html/reg_faqs.html .

[12] FinCEN (2013), *Bank Secrecy Act Requirements - A Quick Reference Guide for MSB*, http://www.fincen.gov/financial_institutions/msb/materials/en/bank_enc e.html .

[13] Office of Foreign Assets Control (OFAC) (2013), *Designated Nationals List (SDN)*, http://www.treasury.gov/ofac/downloads/ctrylst.txt and http://www.treasury.gov/ofac/downloads/t11sdn.pdf .

[14] M. Mehmet, Money Laundering volution Detection, Prevention and Transaction Scoring, PhD Dissertation, George Mason University, 2013.

[15] Hawala, http://en.wikipedia.org/wiki/Hawala, refereed on 10/13/13.